2

CECOM

CENTER FOR SOFTWARE ENGINEERING

ADVANCED SOFTWARE TECHNOLOGY

Subject: **Final Report - Distributed Issues for Ada Real-Time Systems**

CIN: **C02 092LA 0013 00**

23 July 1990

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*

2. REPORT DATE
23 July 1990

3. REPORT TYPE AND DATES COVERED
Final Report

4. TITLE AND SUBTITLE
Distributed Issues for Ada Real-Time Systems

5. FUNDING NUMBERS
MDA 903-87- C- 0056

6. AUTHOR(S)
Thomas E. Griest

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lab Tek Corporation
8 Lunar Drive
Woodbridge, CT 06525

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army HQ CECOM
Center for Software Engineering
Fort Monmouth, NJ 07703-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
STATEMENT A UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This work addresses an approach to reduce the complexity of distributed systems by extending the standard Ada tasking model to handle the distributed processing and by introducing a failure model for reliability issues. It uses software developed in a previous CECOM research project, the Border Defense System (BDS) distributed demonstration application, that addressed the performance benefits that could be gained by the distribution of real-time Ada systems. To achieve increased performance, a new approach to improving parallel execution was studied. The approach was to create a data structure consisting of an array of tasks and distribute the elements of the array across a set of processors. Performance benefits are then achieved as a function of the available processors. A simple failure model appropriate for a class of applications which can tolerate interruptions in service for up to one second was introduced.

14. SUBJECT TERMS
ADA, REAL-TIME, DISTRIBUTED SYSTEMS, FAULT TOLERANCE

15. NUMBER OF PAGES
309

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

DISTRIBUTED ISSUES PROJECT

FINAL REPORT

PREPARED FOR:

U.S. Army HQ CECOM
Center for Software Engineering
Advanced Software Technology
Fort Monmouth, NJ 07703-5000

PREPARED BY:

LabTek Corporation
8 Lunar Drive
Woodbridge, CT 06525

DATE:

13 July1990

EXECUTIVE SUMMARY

The use of distributed systems to obtain gains in system throughput and reliability is expected to continue for the foreseeable future. These systems provide substantial benefits in performance and fault tolerance at the expense of significantly increased complexity. This work addresses an approach to reduce the complexity of these systems by extending the standard Ada tasking model to handle the distributed processor and reliability issues. It uses software developed in a previous CECOM research project that addressed the performance benefits that could be gained by the distribution of real-time Ada systems.

The Ada tasking model is already reasonably well understood from a uniprocessor point of view but it is currently not defined to support characteristics of distributed systems. However, by taking advantage of the same model for distributed systems, fewer new concepts need to be introduced. This reduction in concepts results in a less complex and more flexible system. To achieve increased performance, a new approach to improving parallel execution was studied. The approach was to create a data structure consisting of an array of tasks and distribute the elements of the array across a set of processors. This task array provides the ability to achieve performance benefits as a function of the available processors.

The Ada tasking model is currently silent on failure semantics. This study introduces a simple failure recovery mechanism appropriate for a class of applications which can tolerate interruptions in service for up to one second. Enhanced fault detection and recovery logic have been added to demonstrate the ability to continue operation in the presence of some hardware failures. To improve the flexibility of the system configuration, a new runtime interface has been established to allow dynamic reconfiguration at runtime.

Table of Contents

Table of Contents

Distributed Issues Final Report

1. Introduction

This paper describes the results of a project to investigate issues in distributed Ada programs. It is based on an earlier demonstration project done by the U.S. Army, Center for Software Engineering at Fort Monmouth, NJ[1]. It goes beyond that initial work in two important areas: using task arrays to expand parallelism; and improving the benefits of program distribution beyond performance gains by providing support for fault tolerance.

The earlier project had statically allocated tasks to a processor, and each task was unique, based on a uniprocessor design. The new approach of using task arrays supports the capability to spread the execution of independent iterative operations across a distributed system. The distributed runtime was re-implemented to provide much more flexible reconfiguration capabilities. This made it possible to measure performance gains over a wider set of configurations. The increased flexibility also made it possible for the system to reconfigure under fault conditions and continue operation in a degraded mode.

This paper briefly outlines the application and its key performance characteristics used to demonstrate use of the distributed Ada techniques. Improvements to the application are mentioned, as well as enhancements to the distributed runtime system. Performance benchmarks were done to assess the benefits obtained by distribution of a single Ada program over a network of three Intel 80386 processors connected by an Ethernet. The results of the benchmarks were analyzed to explain how the system reacted under various processing loads and system configurations. These findings were further analyzed to identify design considerations which can be used to improve performance for other distributed applications.

2. Complexity of Distributed Systems

Systems which are distributed tend to be much more difficult to develop than a single processor implementation of the same problem. This is due to several contributing factors. True parallelism creates new scheduling problems and new ways in which resource contention can occur within systems. Distribution introduces the significant new requirement of communication. This inter-processor communication is often at a much higher data rate and less predictable than typical communication with external devices. For this reason, buffering and synchronization become critical design considerations.

Adding to the problem of distributed real-time systems is the issue of maintaining a common sense of time among all of the processors. Application software often becomes involved in the process of keeping track of time and adjusting for time differences among processors. This burden generally does not exist in uniprocessor applications.

Finally, distributed systems generally are constructed in an ad-hoc manner, creating a vast number of dissimilar architectures. This effectively eliminates the mass market necessary to encourage tool developers to support distributed systems. The result is that tools for developing distributed systems are usually custom made and frequently lacking in capability. Horror stories of developers using dozens of separate in-circuit emulators, each with their own console to test distributed systems, are common. Thus the nature of distributed systems simply adds to the difficulties of understanding and solving the complex problems which arise during development.

The benefits of distributed systems are found in essentially three fundamental characteristics: performance, physical separation, and fault tolerance. It is nearly always the case that the price/performance ratio of computer hardware favors many lower cost processors over a single higher performance processor. Furthermore, the ability to handle interrupts by distributed processors reduces the number of context switches (and associated overhead) required. This often results in better response time and overall performance by having several lower cost processors concentrate on a single task rather than having one high cost processor switching among many tasks.

The physical separation of processors provides the ability to have processing resources in close proximity to isolated system hardware. This is often necessary to reduce the wiring and provide a degree of autonomy between subsystems. This can be extremely important in situations which may suffer from localized damage. Similarly, the additional processors make it possible for continued operation in the presence of processor failures. Eliminating all situations where a single component failure will result in system failure is a common axiom of fault tolerant systems.

The growing number of distributed systems in use is evidence to the fact that the benefits of these systems are significant even though there is additional complexity. Techniques that reduce the complexity of these systems would further enhance their attractiveness and are likely to lead to more cost-effective systems and widespread use of fault tolerant computing.

3. Use of the Ada Tasking Model for Distributed Systems

Several approaches are used to support cooperative processing on distributed systems. Most frequently, a message-based mechanism of communication is defined to allow a program on one processor to interface with programs on other processors. These messages are generally developed in an application-specific way with a wide variety of characteristics and requirements. Other techniques include some type of formalized remote procedure call approach with surrogate tasks to execute the procedures that are remotely invoked.

The approach used on this demonstration was to utilize the Ada tasking model of concurrency for all local and remote communication between parallel threads of execution. This approach to concurrency is often referred to as "Distributed Ada" and has a number of advantages to other approaches. Among them are:

1) The ability of the compiler to check interfaces between physical processors.

2) A consistent approach to parallelism - all concurrent activities are expressly stated with a consistent formal mechanism making the system less complex.

3) Re-configuration is facilitated, since the interface between communicating tasks on a processor is the same as that among separate processors, thus allowing tasks to be migrated more easily.

4) Consistency helps to make distributed testing and debugging more easily supported by compiler implementers. Ad hoc approaches make debugging tools prohibitively expensive and generally not as complete.

By utilizing the Ada tasking model, the underlying details for remote communication and maintaining a consistent sense of time are hidden. Since there is a stable model to support, this also creates the possibility for system vendors to provide hardware that is optimized to support distribution of Ada tasks.

The benefits of providing a well understood, uniform approach to concurrent programming should not be underestimated. The ability for developers to have a clear understanding of how their distributed system interacts is essential in lowering the costs and improving the reliability of these systems. From this point of view, using the Ada tasking model for distributed communication and synchronization provides the best opportunity for consistency when programming in the Ada language.

4. Responding to Failures in a Distributed System

The potential to tolerate processor failures is a side-effect of having a distributed system. Frequently this potential is not realized due to the complexity of supporting the detection, isolation, and recovery mechanisms required for fault-tolerant processing. However, safety or mission-critical applications require fault-tolerance and therefore must accept the additional complexity.

The requirements for fault-tolerance can vary from system to system, and the corresponding implementation to support those requirements is substantially different. One of the critical factors is the time in which operations are allowed to be interrupted. Systems that cannot tolerate any interruption in service must perform calculations redundantly and decide which results to accept. More typically, systems are allowed to fail for a few seconds providing that they can come back in service correctly. For these systems, migration of services from failed processors to operational ones is often sufficient to maintain acceptable performance.

The degree to which information is lost (and/or corrupted) during failures also impacts the architecture of the system. The use of stable storage techniques to prevent loss of data is a common approach to continue in the presence of processor failures. This approach checkpoints data to a stable storage area (usually made from redundant memory modules) which is accessible from other processors. If one of the processors fails, another processor can generally carry on from the last checkpoint made by the failing processor. Obviously the amount of time lost due to a failure has a direct relationship on how frequently checkpoints must be made. On the other hand, preventing corruption of data depends on detecting the faults early and preventing the errors from propagating into other portions of the system. This technique is often referred to as establishing "fire walls".

Most failures can be detected by comparing the results of redundant operations, the use of check codes in data, or by using timers to insure that operations complete in their required times. Depending on the type of fault and its detection scheme, the recovery may be as simple as selecting the most likely value based on a majority vote of redundant computations; or it may be a complex process of retries and judgments made on confidence levels in components associated with the failure. The diversity in fault-tolerance requirements and the associated techniques to support them precludes a standard approach to fault-tolerant applications. Instead, flexibility for designers is necessary to allow the method of support to closely match the requirements. For this reason, the demonstration system includes the ability to have application software interface to the configuration control software. (More information on how faults are detected and handled in the demonstration system is provided at the end of section 5.2.) It is clear that many applications will require the ability to have the logic to support fault-tolerance shared between application-specific software and general fault-tolerance software in the runtime.

5. Demonstration Application

To adequately demonstrate the effective use of distributed processing, a real-time application was required to provide a test case program. A synthetic application titled the "Border Defense System (BDS)" which combines target tracking, rocket guidance, and graphics was developed to provide a suitable real-time test. A simulator was included to provide rocket and target motion.

The main characteristics of the BDS are summarized below:

- Hard Deadline Driven application: failure to meet timing requirements will result in mission failure.

- "Processor in the Loop" flight control with dynamic target tracking.

- Complex problem, with interac . among several different functional areas:

  Message Reception (from Sensor Interface and Airborne Rockets)

  Multiple Target Tracking and Prediction

  Multiple Rocket Tracking and Guidance

  Real-Time Graphics Updates

  Real-Time Operator Interface (peak data rate of 500Hz)

- Using current technology: 32-bit Microprocessors (80386-16MHz)

- Initially a separate program was designated for the simulator, however it was temporarily incorporated into the system as additional tasks and placed on a separate processor using the distribution technique.

- All application concurrency is expressed using the Ada Tasking Model (Rendezvous) exclusively.

- The program consists of approximately 6700 Ada LOC contained within 51 compilation units. A copy of the BDS source code is provided in Appendix A.

- The distributed runtime is implemented with 5242 assembly language statements (for compatibility with the vendor runtime) contained in 10 modules. A copy of the distributed runtime source code is provided in Appendix B.

All calculations for both the rockets and targets are done in three dimensional space, however the target simulator currently maintains a constant altitude (Z=0) for the target motions. Each of the aimpoint calculations are computed every 100ms for all of the rockets in flight. On the 16MHz Intel 386 processor the computations currently require approximately 6ms per rocket.

5.1 Enhancements to the Demonstration Application Software

In addition to the fault tolerance and distributed processing capabilities, the demonstration software was enhanced in two ways. First, the rocket simulation algorithms were made much more realistic (and therefore complex). Second, the flight control system was redesigned to be oriented towards a realistic feedback system, that is, the software adjusts the rocket flight based on the effect of previous flight control commands. Previously, the rocket simulator made instantaneous flight corrections rather than corrections based on normal accelerations. This allowed a guidance routine that simply aimed the rocket at the target. A side effect of the feedback approach is an increased sensitivity to (ie. lack of tolerance for) incorrect tracking of the rocket motion. Errors can occur during overload situations where rocket reports can be lost. When this does happen, the rockets become unstable and their flight paths become very erratic.

Accuracy was improved by utilizing 32-bit fixed point types throughout most of the trajectory calculations rather than 16-bit fixed point. Custom fixed point routines were developed that provided substantially better performance than those in the native runtime system which were designed for a 16-bit machine. The rocket guidance equations now utilize 3rd-order processing, which is required to provide the desired accuracy. To provide some insight into what processing is done for rocket guidance, the following computations are performed for each rocket update:

1) The relative (closing) velocities and accelerations of the rocket/target pairings are computed and an estimated impact point is predicted.

2) The rocket's desired velocity vector is then computed and compared to the current velocity vector.

3) Based on the velocity differentials, the desired acceleration is compared to the current rocket acceleration. This provides a desired change in acceleration which is then used to determine the adjustment required in the rocket's attitude.

4) The flight profile is smoothed by integrating the attitude adjustment over a period, which is computed as a function of estimated "time-to-impact". This reduces "overshoot" of the glide slope and had a major beneficial effect on the rocket accuracy.

5.2 Improvements to the Distributed Runtime

Several major changes were made to the underlying distributed runtime. In general, the changes can be classified as enhancements to the configuration flexibility of the system. In particular, the system now is capable of dynamically altering the configuration during system execution. Figure 1 (on the following page) shows the subsystems that make up the entire program which is replicated on each processor. It is shown as being layered from the top application code down through successive levels of abstraction. Conversely, control is passed up through the levels towards the application. Upon initialization, control is given to the System Configuration Setup & Control module which passes control to the distributed runtime. It in-turn transfers control to the vendor runtime which elaborates and activates the application software.

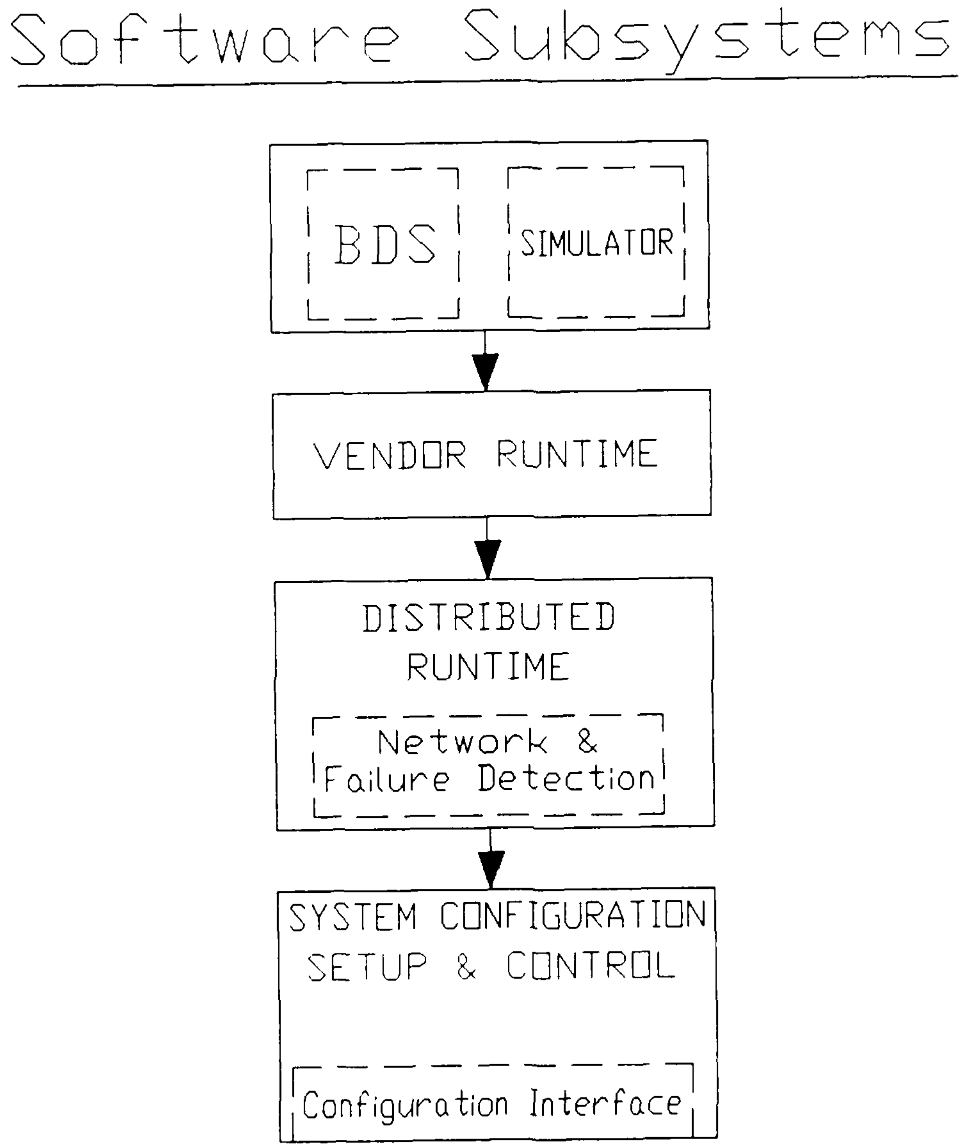

Figure 1. Software Subsystems

Application subprograms request service from the vendor runtime which passes the service request to the distributed runtime. Using a task directory built by the System Configuration Setup & Control module, the distributed runtime determines if the service involves distributed resources. If not, control is returned to the vendor runtime. If the service does involve distributed resources, the distributed runtime carries out the service using its own tasking primitives and the Network Services module.

The Network & Failure Detection module is capable of detecting communication errors or the apparent loss of a processor by using timers with acknowledgement messages and a "heartbeat" mechanism. This mechanism monitors activity from each of the processors. If no activity occurs within a specified period or an acknowledgement is not returned in time, a failure condition exists and failure recovery is initiated. Failure recovery essentially stops application processing and returns control to the System Configuration Setup & Control module.

5.2.1 Distribution Control Details

Each processor initializes the underlying hardware based on what is available in the machine. The Ethernet hardware contains a unique network station address (in Read-Only Memory) which is then used in a table look-up to determine the logical processor ID. The processor ID is then used to determine which processor is the Master (controlling) CPU and which arc the Slaves. The Master is distinguished from the Slaves in that it is responsible for the distributed system Configuration Interface and the system-wide synchronization during start up.

The Master processor prompts the user through a menu system which allows configuration of system parameters. These parameters include:

the maximum number of rockets;

the maximum number targets;

which (of four possible) configurations to use; and

enabling automatic reconfiguration.

If automatic reconfiguration is enabled, a delay may be selected to allow a user to see the error condition for five seconds.

Most of this information is made available to the application via a distributed runtime interface package. For example, it is possible for an application routine to determine if it is running on the Master CPU by testing a boolean variable in the interface. This information can be used by the main program which is activated on all processors to perform machine specific operations. In the case of the BDS it is used to control initialization of the operator interface which only runs on the Master CPU. In addition, the number of Rockets and Targets, and the size of the configuration specific task array is made available. In some sense, these variables can be thought of as parameters to the main program, similar to many host-based Ada program invocations where the command line is provided to the application program as a parameter to the main subprogram. One important distinction from main program parameters is that the runtime package interface is available during the elaboration of other application packages. This allows the size of non-static objects, such as the constraints of the task array, to be based on configuration information.

The configuration selected during setup determines two important aspects of the distributed system: where the tasks are to be resident and how many tasks are to be allocated in application task arrays. Since the same code is present on all processors, a directory is used to determine where they are to be located. The tasks are effectively made remote by suspending them during their normal activation process. A future enhancement could be to add a "self-sizing" mode that would have the master processor search for available processors on start up and after failures and utilize as many as are available.

6. **Performance Characteristics**

The BDS system has been tested to execute on one, two, and three processor configurations. When distributed onto two processors, the simulation tasks run on the second processor. When the third processor is added, the size of an unconstrained task array is increased from one to two, and the second task element of the array is located on the third processor. These tasks in the array divide up the work load of computing the rocket guidance equations.

The performance of the system under different processing loads was studied by collecting timings which reveal the ability of the system to meet the 100ms deadline for computing the next rocket guidance command. A significant portion of this computation is the time it takes to compute the individual aimpoints. This was measured to be approximately 6ms, but varies based on the actual values of the variables in the equations. The variation is due to the algorithm used for square-root which is iterative and will terminate when the current value is known to be within an error bound; and because the multiply and divide machine instructions vary in execution time.

Two important aspects of the timing analysis are the relationships among tasks and the ability to achieve performance gains even with these inter-dependencies. To illustrate this point, a brief description of the sequence of activities for rocket control must be presented:

1) Each rocket control cycle starts with the reception of new rocket flight information. This arrives during a rendezvous with a report buffer task which relays the information from the simulator. Normally, the control task is suspended while waiting for the buffer task to rendezvous, indicating the presence of a new rocket report. When the report arrives, it is provided to the control task and it begins the cycle of computation.

2) The first part of the computation is a correlation step where the current report is correlated to previous reports to create a tracking history. New rocket launches are detected and their histories are initialized. Also, rocket detonations are detected and they are marked as destroyed. Current counts of active and destroyed rockets are maintained and passed on to a status task which updates the screen statistic values. During the correlation processes, a "move" list is generated for updating the rocket symbol positions on the display.

3) Once the correlation has completed, the guidance tasks are given the rocket and target histories which are used to generate the new trajectory data. The guidance tasks are then allowed to run through the trajectory calculations to produce a new aimpoint for each rocket.

4) The control task continues to run in parallel with the guidance tasks after providing them with the information they need. It takes the "move" list generated during the correlation and provides it to the display task.

5) The control task then searches the target list to select the next ideal target if the automatic firing mode is selected and there is an available rocket. (The automatic firing mode indicates that the BDS is to select the next target rather than having the operator select the next target.)

6) The control task then awaits completion of all guidance computations. When the new aimpoints are provided to the control task, it then rendezvous with a guide buffer task which relays the guidance message to the rocket simulator. This completes the timed cycle of interest.

Note that the guidance tasks form a task array which is only one element in size for the one and two processor configurations, but expands to two tasks in the three processor configuration. This distributes the work load of aimpoint computations among two processors and allows the system to support additional simultaneous rocket flights without missing the measured 100ms deadline.

6.1 Benchmark Results

All measurements were taken with 40 active targets. During normal conditions, rocket accuracy was observed to be nearly 99 percent; that is, around one (1) target missed for every 100 rockets expended. In overload conditions where the deadline was missed, rocket accuracy dropped to nearly 0 percent resulting in every rocket missing. The BDS consisted of 11 conventional tasks, an unconstrained array of guidance tasks, and the main program. The entry calls made between the tasks are shown in Figure 2. For specific details of system operation, refer to documentation included in the application source code which is provided in Appendix A. A general description of the tasks are provided below (in decreasing priority order).

In_Char task	Accepts input from the mouse device (mouse interrupt task).
Save task	Buffers mouse data for controlling reticle updates.
Display task	Performs all graphics display updates.
Track_Data task	Buffers target position information between the target tracker and the rocket control task.

Report_Buf task	Buffers rocket position reports from the simulator to the rocket control task (part of the simulator).
Guide_Buf task	Buffers rocket guidance commands from the rocket control task to the simulator (part of the simulator).
Rock_Sup task	Implements the rocket flight simulation (part of the simulator).
Targ_Sup task	Generates and moves simulated targets (part of the simulator).
Control task	Provides overall control for rocket monitoring and flight updates.
Guidance task	Called by the rocket control task to compute flight guidance aimpoints (this is an array of tasks).
Track task	Produces target tracking information for the display and rocket control tasks.
Update task	Updates the statistical status information on the screen.
Bds procedure	Main program used to initialize system operation.

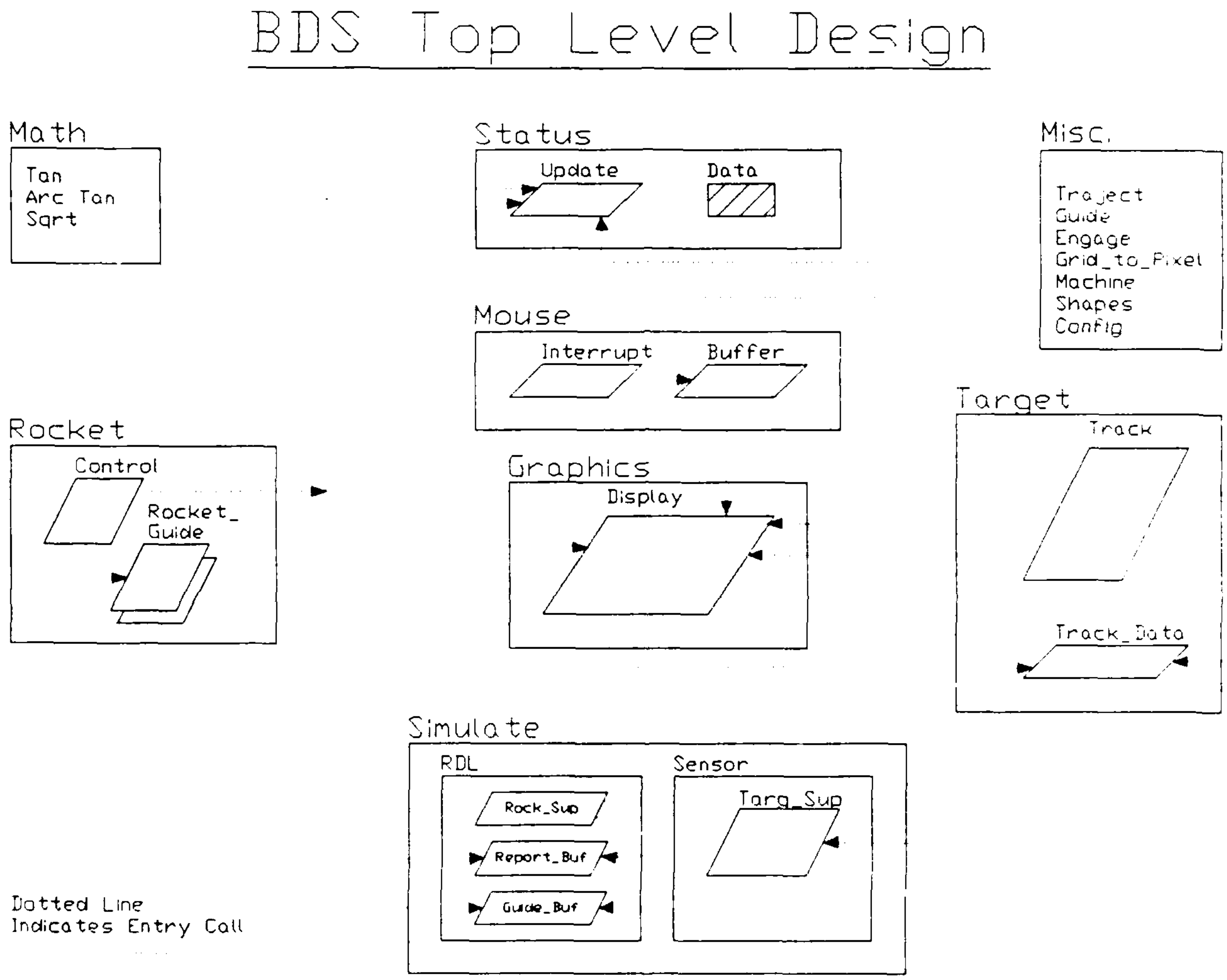

Figure 2. Top Level BDS Design

The measured time is the period that begins when the rocket report becomes available to the control task and ends when the guidance message is provided to the simulator. The allowable time for this has been established as 100ms based on the rocket update characteristics. Successive measurements were taken increasing the number of rockets until either the deadline was missed, or until the number of rockets reached 20. The tested configurations are as follows:

1 Processor	: all tasks are resident
2 Processors	: the simulator tasks are on the 2nd CPU
3 Processors	: the simulator is on the 2nd CPU, the second guidance task is on the 3rd CPU

# of Processors	# of Rockets	Time to complete 100ms Cycle
1	5	[illegible] ms
1	10	105 ms
2	5	38 ms
2	10	77 ms
2	15	118 ms
3	5	41 ms
3	10	73 ms
3	15	79 ms
3	20	90 ms

Extrapolated Saturation Points for Each Configuration:

# Processors	# Rockets
1	9
2	13
3	22

Several observations on the benchmark results can be made. Because the simulator does not participate in the rocket control calculations, and the system is lightly loaded during the control of up to 5 rockets, there is no performance benefit in increasing from 1 to 2 CPUs with only 5 rockets to control. When the number of rockets rises to 10, the dual-CPU system shows a substantial benefit over the single-CPU configuration because both the control computations and simulator computations increase as a function of the number of rockets.

The dual-CPU configuration misses its deadlines when more than 13 rockets become airborne. This is roughly a 44% gain in the number of rockets supported with the single-CPU configuration. The three-CPU configuration is expected to miss its deadlines when more than 22 rockets become airborne. This is roughly a 144% gain over a single-CPU and a 69% gain over the two-CPU configurations. The performance benefits are not linear because only segments of the application are being distributed, and because precedence relationships among the tasks restrict full processor utilization.

Not measured in the benchmarks is the performance of lower priority tasks which are much more substantially impacted during increased processing load. The low priority tasks are the first to relinquish the processor and therefore will suffer first in overload conditions. For example, the task responsible for updating the display statistics begins to starve when only 3 rockets are launched on the single-CPU configuration. In comparison, it continues to run (although at lower frequency) even during overload conditions on the dual and triple processor configurations. The extended life of the lower priority tasks is primarily due to the additional "background" cycles available on the multiple CPU configurations.

It is also due to the I/O blocking time during network communications. That is, while the high priority task waits for access to the network resource, it is blocked from execution which allows lower priority tasks to run. This is one interesting aspect of a distributed system that is not present in shared-memory multiprocessors or single processors. This I/O blocking time has the effect of transferring some execution time directly from high-priority tasks (performing network communications) to other tasks on the system. To the extent that the

I/O blocking time exceeds the processing time required by middle priority tasks, or if the middle and high priority tasks block in contention for the network, then very low priority tasks can be allowed to run. This phenomenon is observed in the demonstration project by the low priority task updating the status display even while high priority tasks miss their deadlines in the multiple processor configurations. The I/O blocking time during rocket "get_report" and "put_guidance" communications is estimated at under 2ms per 100ms cycle. This 2% CPU time is sufficient to prevent starvation in the low priority tasks.

6.2 **Performance Gains**

A common objective discussed for parallel systems is to achieve "scalable" increases in performance. This term implies that when processing resources are added, the "useful" processing increases by a constant factor of the number of processors. For example, for a scale factor of 0.9, then if five times the processors are used, this will result in 4.5 times the useful processing that will be accomplished. This type of measurement is typically applied to computations where the time the computation takes place is only relevant because someone is waiting for the final output of a very large set of computations. However in real-time systems, consistent meeting of short-term deadlines is the measure of performance. For this reason, the conventional sense of "scalable" performance is not totally appropriate.

A real-time system that always meets its deadlines with one processor will not perform better when adding additional processors. It is much more typical that a basic accuracy specification must be met and no benefit is gained by exceeding it. In these cases just meeting the deadline is as good as meeting the deadline in half the required time. The only benefit is the excess capacity which allows future expansion. However, there is a class of algorithms that increase their accuracy based on the amount of time available to execute which might benefit from additional processing resources. So except for reliability concerns and special purpose algorithms, performance can and often does get worse because of communication overhead.

On the other hand, when processing demands limit the capacities of an embedded system, there is potential for substantial benefits to be gained by using additional processors. This is because there are often fixed and variable components to the processing required for system operation. In the BDS, fixed components include the status update, rocket control and report message formatting and transmission, sensor message reception, graphics reticle motion, and runtime overhead (primarily context switching). All of these operations did not vary with the number of targets or rockets supported. Since the additional processing is applied to the variable portion, the increase in system capacity can greatly exceed the increase in processing resources. We refer to this increase in performance as "leveraged performance". In particular, if a system is 70% utilized due to fixed processing requirements, only 30% remains to perform the functions identified as the principal system mission. This 30% can be highly leveraged by adding an additional processor to achieve a nearly 300% increase in system capacity. This characteristic is not clearly evident in the demonstration benchmarks because the tests were taken without the user interface being active and because the target support processing is quite low. The user interface includes moving the reticle which can increase system load by up to 20%, but without a mouse simulator to provide constant simulated motion, it was decided to test the system without the effects of the user interface. The result is that the fixed processing for the demonstration benchmarks was rather low.

The benchmark results are also somewhat biased by a decision to locate only the simulator tasks on the second processor. This resulted in considerable idle time on the second processor which could have been used to do additional rocket calculations. The reason for the decision was to insure the accuracy of the simulation by making sure that nothing interfered with the simulator's processing. A similar effect was created on the three-CPU configuration by choosing a processing balance between the two guidance tasks that was not optimal. Ideally, the two distributed computations would complete at roughly the same time, but the balance selected caused one to finish earlier than the other. The incorrect balance was made because of uncertainties in all of the timing factors that make up the processor loading. Further work is necessary to facilitate an automatic method of determining the optimal configuration for performance.

Thus this study identified two major beneficial factors when considering the use of additional processing resources:

1) When there is a large processing requirement for fixed overhead operations and the addition of processing resources can be applied to the mission-specific processing, which is otherwise limited by the available resources, there can be a leveraged benefit to the system capabilities.

2) When there is the potential for several independent sets of calculations to be performed, the performance increase can be effectively spread over a moderate number of CPUs, limited by the capacity of the network.

The ability to scale-up to a very large set of processors was not investigated by this project. It is believed that the current network architecture would severely restrict such a scale-up due to network contention.

7. Design of Systems Using Distributed Ada

7.1 Hardware Considerations

There are several aspects of system design that impact the utility of the system. Experience has shown that the implementation of distributed systems must be done with the expectation that the hardware will provide reasonable support for fundamental services. These services include:

1) The ability to transmit urgent information in a timely fashion. This requirement usually implies that messages can be prioritized and that the priority is observed in any situation where the potential delay exceeds the allowable allocated time.

2) The ability to broadcast, and later determine that all recipients obtained the message.

3) Sufficient hardware buffering support so that incoming messages will not overrun, resulting in the need for re-transmission.

4) Error detection (or correction) to provide indication of successful message transfer immediately (i.e. within 10 us) by the hardware.

5) The ability to synchronize among the processors.

To a large extent, the network topology has an influence on the real-time response and priority of services provided. The most common network topologies include: Rings, N-dimensional Hypercubes, and Buses. This demonstration project utilized a standard bus topology network - Ethernet, because of its availability and cost. As discussed in a prior report [1], Ethernet is not ideal for real-time use since there is no provision for hardware generated acknowledgements and because access to the bus is granted on a contention basis. The contention-based access method is a poor choice for real-time because no provision is explicitly provided for priority, and two nodes trying to access the medium simultaneously result in a collision. When this occurs, they both wait a random period and retry later. The effect of this on congested networks is a queuing order that is often first-in, last-out. This results in very high worst-case response times which makes meeting fixed deadlines difficult. However, in the demonstration project contention was low since bus utilization was kept below 2%. A more appropriate network for real-time would be either a point-to-point interconnect or a star topology that provided guaranteed response time to all network nodes.

7.2 Software Considerations

The ability to use distributed Ada depends on being able to separate program execution into tasks that can execute in parallel. For embedded applications, usually a small set of independent tasks are naturally present due to their interaction with external objects. These objects (like a video display, operator keyboard, or rocket) are independent and operate concurrently in the real-world. Their control or monitoring therefore naturally maps to separate tasks. The order of execution for these tasks is often asynchronous and is dictated by external events. This type of concurrency is referred to as "natural parallelism". Parallel execution can also be performed on any independent set of operations that are not ordered. Typically these are done as sequential or iterative processes because treating each operation as a separate task would result in additional context switches. However, applying multiple processors to the computations can more than compensate for the small overhead of the additional context switches. This forced parallelism can help to increase the amount of processors which can be effectively used.

As a design goal, system designers should attempt to reduce interdependency of tasks as much as possible. Usually this requires detailed knowledge of the application and implies being able to partition system functions into tasks that have a high degree of autonomy. Tasks (other than monitors for shared access control) should not be used for activities that cannot be done in parallel.

Secondly, avoid serialization in the order of task synchronization if possible. For example, if task A must wait for both task B and task C to provide data, it should not enforce the order of which task it must rendezvous with first unless they are always guaranteed to arrive in a particular order. Instead, a conditional rendezvous should be used to prevent unnecessary serialization of events.

Third, tasks which must reside on a particular processor because of required access to hardware interfaces should provide the minimum service necessary to support efficient use of that hardware. This approach provides additional design freedom to locate a greater percentage of the required processing demands according to communication demands rather than specific hardware dependencies.

Finally, the software should be designed to operate correctly during overload conditions. This requires the ability to detect the overload condition, maintain consistency of data, and (ideally) to support the dynamic load shedding necessary to make good use of the available processing resources. This last provision is absolutely necessary to continue operation in the presence of hardware failures, since the loss of processing resources will almost certainly increase the likelihood of overload conditions.

7.3 Differences Between Distributed and Uniprocessor Implementations

There were two major unexpected differences identified between execution on a distributed system and a uniprocessor system. The first was mentioned in the performance section and involves the impact of I/O blocking on high priority tasks. This prevents starvation in low priority tasks and allows the status display and target tracker tasks to continue to run even under conditions of very high utilization. This effect was due to the desire to run tasks as often as possible during non-overload conditions and yet allow the more critical tasks to obtain the CPU during overload. This may have been avoidable using a more complex algorithm to schedule the status and target tracker tasks based on the available time for them to run.

The second difference was due to a design assumption that made the program erroneous. During a prototype enhancement of the application, a change was made in the program to use message sequence numbers in order to detect the loss of successive reports (due to buffer overwrite during overload conditions). Although the code was made obsolete by another function and was essentially removed prior to system integration, a seemingly harmless portion remained which examined the rocket report message. This message was a task entry "**out**" parameter from the simulator's report buffer task and it was examined to detect sequence numbers that changed from report to report. By convention, only the first "N" items in the message are considered valid, where "N" is provided at the beginning of the message. However the sequence monitoring code did not examine this count before testing the sequence numbers and assumed that the first rocket position was always valid. In the single-CPU configuration, the entry call "out" parameter was passed "by address", and the buffer task only updated those records that were active. In distributed configurations, entry call parameter passing must be done as "copy-in, copy-out" so the data can travel over a network and it operates without regard to the contents of the objects being copied. The result is that all of the records are updated during each entry call. This had the effect that the sequence software became confused and rejected messages due to apparent bad sequence numbers. The root of the problem was due to a dependence on the parameter passing mechanism used for entry parameters. RM 9.5(6) states that:

> "The parameter modes for parameters of the formal part of an entry declaration are the same as for a subprogram declaration and have the same meaning (see 6.2)."

RM 6.2(5) defines **out** mode parameters as:

> The formal parameter is a variable and permits updating of the value of the associated actual parameter.
>
> The value of a scalar parameter that is not updated by the call is undefined upon return; the same holds for the value of a scalar subcomponent, other than a discriminant. Reading the bounds and discriminants of the formal parameter and of its subcomponents is allowed, but no other reading.

RM 6.2(7) continues:

> "For a parameter whose type is an array, record, or task type, an implementation may likewise achieve the above effects by copy, as for scalar types. In addition, if copy is used for a parameter of mode **out** , then copy-in is required at least for the bounds and discriminants of the actual parameter and of its subcomponents, and also for each subcomponent whose type is an access type. Alternatively, an implementation may achieve these effects by reference, that is, by arranging that every use of the formal parameter (to read or to update its value) be treated as a use of the associated actual parameter, throughout the execution of the subprogram call. The language does not define which of these two mechanisms is to be adopted for parameter passing, nor whether different calls to the same subprogram are to use the same mechanism. The execution of a program is erroneous if its effect depends on which mechanism is selected by the implementation."

The essential difference in implementation approach is that when call by reference is used, only those records that are explicitly assigned a value are altered by the entry call. When call by copy-in/copy-out, all of the values are altered. In either case it is considered erroneous to reference a value that is not updated, but in fact the single-CPU application was doing this. There was no effect in the call by reference implementation, but when the task became distributed and call by copy was used, the latent error in the software was activated. The lesson is that erroneous programs are more likely to operate incorrectly on distributed systems because a different parameter passing mechanism is likely to be used for composite objects.

8. Limitations of the Distribution Support

8.1 Task Identification

The current mechanism for identifying tasks is the task's base priority. This was a convenient mechanism to use for a number of reasons. First, it is stored in the vendor's runtime task control block and is therefore available during any runtime call. It can be set via a **pragma** in the source code and therefore a unique ID can be associated with each task type. This approach was selected because it provided an expedient solution to identifying application tasks to the distributed runtime without modification to either the compiler or the vendor runtime.

Future versions would obviously use some other field in the control block because this technique is very limited. One complication is with the handling of task arrays. Since they are of the same task type, all tasks in an array have the same priority. This problem was circumvented by leaving sufficient space between adjacent priorities to change the priority of each task in the array during activation if necessary to make them unique.

Another approach was considered, but rejected because of development time. This approach was to use an intermediate file produced for a debugger to correlate the actual name of each task with the execution addresses where the task type is created and the task object is activated. This information would be combined with the designer's distributed configuration specification and loaded with the runtime. Since this specification would use the actual expanded name of each task, the limitation of having only as many tasks as there are priorities would be removed. During task type creation and task activation, the return address on the stack during the runtime call provides the execution address and could be used to identify the task type and object in the configuration specification. This approach requires the development of a tool to process the compiler intermediate files, linker maps, and the distributed specification file. There may be other problems that are not obvious but the method appears suitable for some applications.

8.2 Code Replication

Currently all of the code for each task must be resident on each processor. In some applications this is unacceptable since there will not be sufficient memory to support the entire application on every processor. One approach to reducing this overhead may be to use subunits for all task bodies and then create a sublibrary with dummy bodies appropriate for each processor configuration. Linking the program with respect to each sublibrary will then produce a load image for that processor configuration. However, some compilation systems do not allow subunits to appear in any library other than the library in which the parent unit was compiled. In any case, it would probably be advisable to have a tool which automatically created the necessary sublibaries and subunit bodies and therefore reduce the chance of error in generating the individual load images.

8.3 Network Error Recovery

Each network message is acknowledged which allows transmission errors to be detected, however there is no provision for re-transmission. The system will simply shutdown and reconfigure on the first error. This is somewhat severe since re-transmission can usually be done without loosing real-time if the acknowledgements are prompt. (Measurements on the BDS indicated typical acknowledgement times of 400 to 800 microseconds.) The only complication is saving the data for re-transmission. Currently, transmissions transfer application variables directly to a single hardware transmit buffer. Once the message is sent, the buffer is reused and the application task is allowed to continue. There is no provision to save the data for later re-transmission. This could be done with very little overhead by allocating additional space in the hardware buffer for transmissions, however the hardware in use is configured with only 8KB (kilobytes) of memory and therefore this is not practical. By expanding the memory to 32KB, or by using system memory and performing an additional copy, the data can be retained until the acknowledgment is received and the buffer can be freed.

9. Compiler and Runtime Problems

Considerable effort was spent isolating problems associated with the Ada implementation. The implementation was an upgrade from the version used on the previous demonstration project and was far more reliable than that earlier version. Nevertheless because code generator errors continued to appear, a decision was made to not use optimization for compilation of many of the units and to greatly restrict use of **pragma** inline. This noticeably improved the reliability of the generated code.

Even with these restrictions, two problems were identified during final integration testing. In one unit where inline was still used, the compiler failed to generate the same (correct) addresses for variables which were initialized during package elaboration. These variables happened to be pointers within a circular queue, and the error would generally go unnoticed if the values of the memory locations happened to be the same. This was typically the case during testing since the system memory initialization routine would zero all of memory to prevent parity errors. However, depending on the contents of memory when the program was loaded the system could crash if the two values were very large or not equal to each other. While single stepping through the program it was noticed that the pointers were being initialized properly which was the obvious expected source of the problem. However, during program execution the system would still crash, and the pointers would have invalid data. Use of the processor's special debug registers to halt on references to data, much like an in-circuit emulator, helped to track down the problem and realize that two different locations were being used for the same variable.

The second problem was related to computation of 32-bit fixed point values. When a small value is divided by a large value resulting in zero, and exactly one of the operands is negative, the runtime would incorrectly assume that an overflow had occurred because the result was not negative. Instead it was zero, which was also a legal value. This problem was fairly quickly resolved since it has been noticed in the earlier release of the runtime. It was fixed by changing the conditional branch instruction to allow for zero results.

10. **Summary**

This project demonstrated that the use of distributed Ada can provide increased performance benefits and fault tolerance for a reasonably complex real-time application. These benefits can take the form of simply using the parallelism natural in the application or by expanding the parallelism using task arrays to compute multiple independent calculations. In particular, the ability to distribute elements of a single Ada object was demonstrated by distributing an array of tasks to divide up the workload among several processors.

Task precedence relationships create considerable design difficulties when trying to analyze a system for optimal parallel operation. New tools and scheduling paradigms are required to assist designers in resolving these difficulties. However, techniques do exist to provide marginal improvements in parallel operation by reducing dependencies and encouraging the judicious use of synchronization primitives. An example of such a technique is the use of buffering schemes and control variables to de-couple tasks. This technique may require detailed knowledge of the application to insure proper execution with the buffering scheme.

The concept of "scalable performance" was discussed, and a more appropriate term for real-time embedded systems: "leveraged performance" was introduced. This concept recognizes the limiting factors in real-time systems, and emphasizes the potential of capacity increase factors greater than one (1) for applications with a substantial portion of processing dedicated to execution requirements of fixed duration.

A very important potential gain of distributed systems is the ability to utilize the natural redundancy in the hardware to achieve increased fault tolerance. Typical embedded systems have had, and will continue to have multiple processors. The problem of reconfiguring the system during failure conditions has prevented widespread use of fault tolerance techniques. Distributed Ada appears to be a good candidate for reducing this problem to a manageable level for applications which should operate in the presence of failures.

Fault tolerance concerns were examined and a clear need arose to provide an interface between the fault tolerant runtime and the application. The application must be able to have some sense of the available processing resources in order to adapt to the configuration.

An example was shown of an erroneous program which failed on the distributed implementation when it had previously run correctly in a uniprocessor configuration. A conclusion was drawn that since distributed systems are likely to use both "pass by copy" and "pass by reference" mechanisms for parameters of composite types, programs which erroneously depend on the parameter passing mechanism are more likely to fail on distributed systems.

Finally, compiler reliability still poses a serious problem when trying to obtain the highest performance possible using complex optimizations and language features such as **pragma** inline. Mission and safety critical applications should consider the impact of having to operate without the use of these performance enhancements.

11. References

[1] CECOM Center for Software Engineering, "Real-Time Ada Demonstration Project", CIN # C02 092LA 000900, Final Report, delivered by LabTek Corp., May 31, 1989.

[2] Reference Manual for the Ada Programming Language, ANSI/MIL Standard 1815A-1983.

12 Appendix A - Border Defense System Ada Source Code

The source code for the BDS system follows in alphabetical order of the unit names (specifications precede bodies).

```
----------------------------------------------------------------
--% UNIT:      Aim_Data package spec.                          --
--% Effects:   Holds Rocket/Target history information for Guide.  --
--% Modifies:  Rocket_Info is global data and is modified by Guide.  --
--% Requires:  Initialization is required and performed by Guide.  --
--% Raises:    No explicitly raised exceptions are propagated.  --
--% Engineer:  L. Griest.                                      --
----------------------------------------------------------------

--|
--| PACKAGE SPEC : Aim_Data
--|
--|  Aim_Data contains the information for Guide necessary to control the
--| rockets in flight.  The data is initialized by Guide when the rocket
--| is taking off from a launch position.  Note that curr_nnnn signifies the
--| most current postion of an object and that last_nnnn signifies the position
--| the object had immediately prior to this interval (assuming no overload
--| condition).  The prev_nnnn field exists only for rockets and represents
--| the position the rocket had two intervals prior to this one.  This field
--| is used to calculate the velocity of the rocket last interval in all three
--| axis.  This information is not needed for targets.
--|  RATE_REC_TYPE is necessary to provide the accuracy necessary when
--| calculating accelerations and velocities, particularly at launch times.
--|

-- Modifications Log
--
-- 89-11-09 : LJG => Original created.
--

with Types; use Types;

package Aim_Data is

  type RATE_REC_TYPE is record
    X : Types.RATE_TYPE;
    Y : Types.RATE_TYPE;
    Z : Types.RATE_TYPE;
  end record;

  type ROCKET_INFO_TYPE is record
    LAST_TARG         : Types.POSITION_TYPE;
    CURR_TARG         : Types.POSITION_TYPE;
    PREV_ROCK         : Types.POSITION_TYPE;
    LAST_ROCK         : Types.POSITION_TYPE;
    CURR_ROCK         : Types.POSITION_TYPE;
    OLD_AIMPOINT      : Types.AIMPOINT_TYPE;
    BOOST_PHASE       : BOOLEAN; -- rocket currently in boost phase?
  end record;
```

```
  type ROCKET_INFO_ARRAY is array(Types.ROCKET_INDEX_TYPE) of ROCKET_INFO_TYPE;

  ROCKET_INFO : ROCKET_INFO_ARRAY;
end Aim_Data;
```

```
-------------------------------------------------------------------
--% UNIT:     Aimpoint function spec.                             --
--% Effects:  Compute new aimpoint based on acceleration requirements.  --
--% Modifies: No global data is modified.                         --
--% Requires: No initialization is required.                      --
--% Raises:   No explicitly raised exceptions are propagated.     --
--% Engineer: T. Griest.                                          --
-------------------------------------------------------------------

--|
--| FUNCTION SPEC : Aimpoint
--|
--|   Aimpoint is responsible for returning a new elevation and azimuth to
--| the caller based on the acceleration adjustment.
--|

-- Modifications Log
--
-- 89-11-6 : TEG => Original Created.
--
with Types; use Types;  -- for operators on types only!
with Aim_Data; use Aim_Data;

function Aimpoint(OLD_AIMPOINT    : Types.AIMPOINT_TYPE;
                  ACCEL_ADJUST    : Aim_Data.RATE_REC_TYPE)
                  return Types.AIMPOINT_TYPE;
```

```
-----------------------------------------------------------------------
--% UNIT:     Aimpoint function body.                                  --
--% Effects:  Compute new aimpoint based on acceleration requirements. --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
-----------------------------------------------------------------------

--|
--| FUNCTION BODY : Aimpoint
--|
--|  The selected aimpoint is a function of the desired change in acceleration
--|  for each of three axis and the current aimpoint.  The following rules are
--|  used:
--|    TO INCREASE Z ACCELERATION adjust elevation towards 16384 (straight up)
--|    TO INCREASE X ACCELERATION adjust azimuth towards 0      (straight right)
--|
--|  Obviously there is some interaction among these components.  Z
--|  is the controlling axis since its acceleration is not dependent on azimuth
--|  and the accelerations in X and Y are dependent on elevation.  Once the
--|  change in elevation has been established, the impact on X and Y
--|  accelerations are computed, then a proper azimuth is selected based
--|  on the above rules.
--|  To implement the friendly fire suppressor only generate and process
--|  elevations between -16384 (straight down) and 16384 (straight up).
--|  and azimuths between 0 (straight right) and 32767 (straight
--|  left).
--|  When adjusting elevation, reduce negative impact since gravity will
--|  have a compensating effect.
--|

-- Modifications Log
--
-- 89-11-03 : TEG => Original Created.
--

function Aimpoint(OLD_AIMPOINT    : Types.AIMPOINT_TYPE;
                  ACCEL_ADJUST    : Aim_Data.RATE_REC_TYPE)
                  return Types.AIMPOINT_TYPE is

  max_climb   : constant  :=  16384;
  max_descend : constant  := -16384;
  left        : constant  :=  32767; -- full left while going forward
  right       : constant  :=      0; -- full right while going forward
  elev_factor : constant  :=  10000; -- controls flexibility in turning rocket
  az_factor   : constant  :=  10000; -- controls flexibility in turning rocket


  NEW_AIMPOINT: Types.AIMPOINT_TYPE;
  ADJUST_ELEV : Types.EXTENDED_BAM;    --   use 32-bit values for intermediate
```

```
  ADJUST_AZ    : Types.EXTENDED_BAM;
  TEMP         : Types.EXTENDED_BAM;

begin
--  Put("In Aimpoint: ADJUST_ELEVATION: ");
--
--  Change elevation to effect Z acceleration first.
--
  ADJUST_ELEV := Types.EXTENDED_BAM(ACCEL_ADJUST.Z * elev_factor);
  if ACCEL_ADJUST.Z < 0.0 then
    ADJUST_ELEV := ADJUST_ELEV / 2;   -- reduce descend angle because of gravity
  end if;
  TEMP := Types.EXTENDED_BAM(OLD_AIMPOINT.ELEVATION) + ADJUST_ELEV;
--
-- Must perform limit check on climb/descend.
--
  if TEMP > max_climb then
    NEW_AIMPOINT.ELEVATION := max_climb;
  elsif TEMP < max_descend then
    NEW_AIMPOINT.ELEVATION := max_descend;
  else
    NEW_AIMPOINT.ELEVATION := Types.BAM(TEMP);
  end if;

--
--  NOW PROCESS AZIMUTH (Using only X, let Y take care of itself!)
--
  ADJUST_AZ := Types.EXTENDED_BAM(-ACCEL_ADJUST.X * az_factor);
--
--  Do limit checks to make sure we don't start turning back towards FLOT
--
  TEMP := Types.EXTENDED_BAM(OLD_AIMPOINT.AZIMUTH) + ADJUST_AZ;
  if TEMP > left then
    NEW_AIMPOINT.AZIMUTH := left;
  elsif TEMP < right then
    NEW_AIMPOINT.AZIMUTH := right;
  else
    NEW_AIMPOINT.AZIMUTH := Types.BAM(TEMP);
  end if;
  return NEW_AIMPOINT;
end AIMPOINT;
```

```
-------------------------------------------------------------------
--% UNIT:     BDS Spec & Body.                                    --
--% Effects:  Initiates main processing, loops recording idle time.   --
--% Modifies: No global data is modified.                         --
--% Requires: Status.Initialize be called before Mouse.Initialize.    --
--% Raises:   No explicitly raised exceptions are propagated.     --
--% Engineer: T. Griest                                           --
-------------------------------------------------------------------

----------------- Distribution and Copyright ----------------
-- Derivation   : LabTek Border Defense System V2.0
--
-- This Border Defense System Software inherits the LabTek copyright.
-- The following copyright must be included in all software utilizing
-- this application program.
--
-- Copyright (C) 1989, 1990 by LabTek Corporation, Woodbridge, CT, USA
--
-- Permission to use, copy, modify, and distribute this
-- software and its documentation for any purpose and without
-- fee is hereby granted, provided that the above copyright
-- notice appear in all copies and that both that copyright
-- notice and this permission notice appear in supporting
-- documentation, and that the name of LabTek not be used in
-- advertising or publicity pertaining to distribution of the
-- software without specific, written prior permission.
-- LabTek makes no representations about the suitability of
-- this software for any purpose.  It is provided "as is"
-- without express or implied warranty.
--
----------------- Disclaimer ------------------------------
--
-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.
-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exist.
--
-- In no event shall any person or organization of people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.
--
------------------END-PROLOGUE------------------------------

--|
--| TASK BODY : BDS main procedure
--|
--|   The BDS main procedure is used to synchronize the start of events within
--| the entire system.  During elaboration until the start of the procedure,
--| the system will settle to a known state.  Then when the call to Status
--| is performed, the statistics titles will be printed on the screen.  After
--| this is performed the Mouse initialization is completed.  Then two
```

```
--| successive entry calls are done.  The first starts the Rocket.Control task
--| going.  The second signals the Track task to begin processing target
--| information.
--|   The "--$TP(NNNN) ..." signifies a Time Point stamp location.  There is a
--| tool built by LabTek which transforms these comments to Ada code which
--| performs a call to a TimeStamp procedure.  In order to keep from filling
--| memory too fast, a loop is used to force the main procedure to loop slower
--| than it normally would.  This time stamp routine will enable approximations
--| of the amount of free time the processor has, since this procedure has the
--| lowest priority.
--|

-- Modifications Log
--
-- 88-09-30 : TEG => Original created.
--

with Config;            -- global configuration parameters
with Status;            -- updates statistics used
with Types;             -- global types definitions
with Mouse;             -- mouse movement and rocket launching
with Rocket;            -- rocket attitude and aimpoint calculations
with Target;            -- generation of various targets
with Interrupt_Control; -- enabling and disabling of (all) interrupts
with Machine_Dependent; -- individual pixel plotting for EGA
with Time_Stamp;        -- run time profiler
with Distrib;

procedure BDS is

-- This is the main program for the Border Defense System.  It has only
-- two calls which are of any importance, i.e., the other code is for
-- timing purposes only.  The first call performs initialization of the screen
-- statistics descriptions and their initial values.  The second call starts
-- the mouse.

use Types;                                 -- for visibility to "+"

pragma PRIORITY(Config.bds_priority);

COUNT : Types.WORD;                        -- these two variables are for
SLOW  : Types.WORD;                        -- slowing the time stamps

begin
  if Distrib.MASTER then
    Status.Initialize;                      -- print screen statistics
    Mouse.Initialize;                       -- must be done after status signal
    Rocket.Control.Start;
    Target.Track.Start;
  end if;
  loop                                     -- done with initialization
```

```
    Time_Stamp.Log(0001);      --$TP(0001) BDS main time stamp
    SLOW := 1;
    for COUNT in 1..2000 loop
      SLOW := SLOW + 1;
    end loop;
  end loop;
end BDS;
```

```
-------------------------------------------------------------------
--% UNIT:     Config Spec.                                       --
--% Effects:  Provides system-wide configuration constants.      --
--% Modifies: No global data is modified.                        --
--% Requires: No initialization is required.                     --
--% Raises:   No explicitly raised exceptions are propagated.    --
--% Engineer: T. Griest.                                         --
-------------------------------------------------------------------

--|
--| PACKAGE SPEC : Config
--|
--|   The Config package (which currently has no body associated with it) can
--| be contrasted to the Types package.  While the Types package is responsible
--| for declaring the various global types used throughout the BDS, the Config
--| package is used to declare global constants.  The rocket launch trajectory
--| is used to determine the azimuth and elevation of the rocket before takeoff.
--| The kill radius is used to determine the explosive power of the rocket as
--| it hits near a target.  If the target is within the radius determined below,
--| then it is considered to have been close enough to the rockets explosion to
--| have caused damage severe enough to render it immobile and harmless.  Note
--| that the BDS does not take into account the case of a rocket doing "some"
--| damage on the target; every target is considered to be totally missed or
--| fully hit.  There are two battle areas that can be considered in the BDS.
--| The first is the battlefield area which is the "real" area of conflict and
--| the other is the screen battlefield which is shown to the user.  The "real"
--| battlefield is used by the Target Tracker, the Rocket Controller, and their
--| respective data links to their sensors.  In order to provide a proportional
--| view of the "real" battlefield area, the number of pixels in X and Y was
--| calculated.  The screen battlefield does not take up the entire screen;
--| some is left for the display of statistics.  The calculations done in the
--| "real" battlefield are three dimensional, those on the battlefield screen
--| two dimensional.
--|   The bytes per storage unit is used for transportability reasons.  A count
--| of bytes required for each tasks' stack (including nested procedures) was
--| included so that the application could be less implementation dependant.
--| To leave defaults in place would require that the largest stack frame
--| be used for all tasks stacks regardless of the actual space needed.  By
--| specifying the amount of stack needed on a per task basis, less memory
--| is used.
--|   The interval constant declared below is the basic unit on which timing
--| in the BDS is performed.  It specifies that an entire iteration (which
--| includes a rocket update, a target update, and a possible mouse or
--| statistics update) all be performed in 100 millisecs.  The delays specified
--| in the timed tasks (Target.Track, Rocket.Control, Simulate.RDL.Rock_Sup and
--| Simulate.Sensor.Targ_Sup are the timed tasks currently) are calculated
--| so that they will wakeup once every interval (100 ms).  The rest of the
--| system derives its timing from these drivers.
--|   The priorities are grouped together here because priorities specified
--| individually in each task declaration does not help anyone looking to
--| determine priorities which are relative to each other.  The Mouse_Buffer
```

```
--| task is not the highest priority task.  Since there is a mouse associated
--| with the system, which uses an interrupt entry call and is treated as a
--| task, it uses the hardware interrupts to determine its priority.  Since
--| a task must always be sitting at the accept to receive the interrupt tasks'
--| entry call, the Mouse_Buffer task (which is responsible for translating
--| the X-Y motion of the mouse (and any buttons pushed) into motion of the
--| reticle) is defined at the highest software priority level.  For the same
--| reason, as well as to be able to keep the screen in real time, the Graphics
--| task is declared with the next available priority.  In order to increase
--| throughput from the simulator to the BDS the buffers which route rocket
--| and target data are declared with the next highest available priority.
--| Because the simulator contains the two tasks which are scheduled according
--| to a deadline (RockSup and TargSup) these tasks are next in the priority
--| line.  Then the rocket controller for the BDS and the target controller
--| for the BDS are (respectively) assigned their priorities.  Below these
--| tasks is the statistics task priority.  It is allowed to be low because
--| of the liberal timing requirements placed on it by the requirements
--| documentation.  Obviously, since the main program performs no function
--| which is of use to the BDS, it is assigned the lowest priority.
--|

-- Modifications Log
--
-- 88-10-11 : TEG => Original created.
-- 89-11-16 : MPS => Added launch attitudes and locations.
--

with System; use System;

package Config is

-- The following two constants allow the space needed for the various tasks to
-- be declared in bytes.
byte                      : constant := 8;        -- 8 bits
bytes_per_storage_unit    : constant := byte / System.STORAGE_UNIT;

-- Now define battlefield area perimeters

meters_in_battle_area     : constant := 4_000.0; -- in X and Y direction

meters_per_X_pixel        : constant :=  9.625;  -- rounded up to nearest
meters_per_Y_pixel        : constant := 11.875;  -- Types.METER.
max_pixels_in_battle_area : constant := meters_in_battle_area
                                              / meters_per_X_pixel;
--
-- Task priorities in order of decreasing urgency.
--
--  NOTE: MOUSE IN_CHAR has no priority because it runs
--  completely at the hardware interrupt level.
--  The idea implemented here is that all the Simulator information is
--  of higher priority than the actual Border Defense System code.
```

```
save_priority        : constant PRIORITY  := PRIORITY'last;       -- Mouse_Buffer
display_priority     : constant PRIORITY  := save_priority-1;     -- Graphics
track_data_priority  : constant PRIORITY  := display_priority-1;   -- Target
report_buf_priority  : constant PRIORITY  := track_data_priority-1;-- Sim.RDL
guide_buf_priority   : constant PRIORITY  := report_buf_priority-1;-- Sim.RDL
rock_sup_priority    : constant PRIORITY  := guide_buf_priority-1; -- Sim.RDL
targ_sup_priority    : constant PRIORITY  := rock_sup_priority-1;  -- Sim.Sensor
control_priority     : constant PRIORITY  := targ_sup_priority-1;  -- Rocket
guidance_priority    : constant PRIORITY  := control_priority-1;   -- Rocket
track_priority       : constant PRIORITY  := guidance_priority-2;  -- Target
update_priority      : constant PRIORITY  := track_priority-1;     -- Status
bds_priority         : constant PRIORITY  := update_priority-1;    -- Main

--
-- define entire hi-res screen display borders.  The screen is divided into
-- two main sections.  There is the battlefield area where the targets, rockets,
-- and reticle are allowed to move, and there is the statistics area where our
-- current statistics will be displayed.  The maximum number of digits allowed
-- in any statistics displayed is statistics_length.  Between the statistics and
-- the battlefield there is a border.
--
--
-- define entire screen constants
--
entire_screen_left        : constant := 0;
entire_screen_right       : constant := 639;
entire_screen_top         : constant := 0;
entire_screen_bottom      : constant := 349;
--
-- define battlefield display borders and center.
--
battlefield_screen_left   : constant := 222;     -- starting (left)
battlefield_screen_right  : constant := 638;     -- ending (in pixels)
battlefield_screen_top    : constant := 1;       -- starting (top)
battlefield_screen_bottom : constant := 338;     -- ending (in pixels)
battlefield_center_x      : constant := 430;     --
battlefield_center_y      : constant := 169;     --
--
-- define border between battlefield and statistics.
--
border_left               : constant := 221;     -- starting (left)
border_right              : constant := 639;     -- ending (in pixels)
border_top                : constant := 0;       -- starting (top)
border_bottom             : constant := 339;     -- ending (in pixels)
--
-- define statistics display borders.
--
status_left               : constant := 0;       -- starting (left)
status_right              : constant := 220;     -- ending (in pixels)
status_top                : constant := 0;       -- starting (top)
status_bottom             : constant := 349;     -- ending (in pixels)
```

```
--
-- statistics_length is the number of digits allowed in any status field, and
-- stats_title_max_length is the max number of letters any particular
-- statistics title may contain.
--
statistics_length        : constant := 4;
stats_title_max_length   : constant := 11;
number_of_titles         : constant := 12;

max_targets              : constant := 50;      -- total targets
max_rockets              : constant := 20;       -- total rockets

interval                 : constant := 0.100;   -- basic interval is 100ms
gravity                  : constant := 9.80665; -- meters/sec**2

--  launch attitude
launch_azimuth           : constant := 16384;   -- straight ahead in BAMS
launch_elevation         : constant := 15000;   -- 7.6 degrees off straight up

launch_x                 : constant := 2000.0;
launch_y                 : constant := 60.0;
launch_z                 : constant := 10.0;

kill_radius              : constant := 10.0;    -- 10 meters x 10 meters

end Config;
```

```
----------------------------------------------------------------------
--% UNIT:     Control task subunit.                                 --
--% Effects:  Provides overall control for rocket flight and display. --
--% Modifies: Updates rocket data base in Rocket body.              --
--% Requires: No initialization is required.                        --
--% Raises:   No explicitly raised exceptions are propagated.       --
--% Engineer: T. Griest.                                            --
----------------------------------------------------------------------

--|
--| TASK BODY : Rocket.Control
--|
--|   The Rocket.Control task controls the information coming in from the rocket
--| support task and the target support task.  With this information it
--| develops a list for the guidance task to work on (the guidance task being
--| in charge of developing new aimpoints for each rocket), updates the
--| statistics, launches a new rocket if necessary, sends the new positions of
--| the rockets to graphics for displaying, receives from guidance the new
--| aimpoints, and delivers those to the Rocket Support task in the simulator.
--|   The purpose of the disengaged pointers, the engage flag, and the rocket
--| launch flag is to support the specification that only one target can be
--| marked destroyed each interval and that only one rocket can be launched
--| per interval.  Also going along with this is that targets can only be
--| created one per interval.  This helps to maintain a better average
--| response time, thus predictibility of the amount of time this routine will
--| take is enhanced.  If a graph was drawn of CPU utilization versus time, and
--| the targets and rockets were all allowed to be created and destroyed in
--| one interval as necessary, then several destroyed rockets and consequently
--| several created targets (the next interval) would appear on the graph as
--| spikes.  It is necessary to eliminate "spikes" from the BDS because it is
--| a deadline driven mission.  For this reason, the Simulate.RDL.Rock_Sup task
--| and the Simulate.Sensor.Targ_Sup task have timing loops surrounding their
--| executable code.  This technique allows for better fault tolerance; if one
--| of the buffer tasks or even one of the four main tasks mentioned above were
--| to be disabled because of an error, the rest of the system would still be
--| able to function properly.
--|   The rendezvous mechanism with the guidance task is done as if there were
--| an array of guidance tasks.  Although there is only one guidance task at
--| present, if more were added and they were on separate processors, this
--| design would facilitate the distribution of those tasks.
--|

-- Modifications Log
--
-- 88-11-10 : TEG => Original Created.
-- 89-11-22 : MPS => History information moved from rocket package body
--                   to rocket.control task body.
--

with Interrupt_Control;
with Grid_to_Pixel;
```

```
with Simulate;
with Target;
with Sync;
with Calendar;
with Engage;
with Time_Stamp;
  pragma ELABORATE(Interrupt_Control, Grid_to_Pixel, Sync,
                   Simulate, Target, Calendar, Engage, Time_Stamp);

separate(Rocket)
task body Control_Type is

  use Calendar;           -- for operators
  use Types;              -- for operators
  use Sync;               -- for operators

  package RDL renames Simulate.RDL;   -- make simulator transparent

  dis_list_size    : constant := Config.max_rockets;

  type HISTORY_REC_TYPE is record
    ROCKET_OLD      : Types.POSITION_TYPE;
    TARGET_OLD      : Types.POSITION_TYPE;
    TARGET_AIMED_AT : Types.WORD_INDEX;
  end record;

  type HISTORY_LIST_TYPE is array(Types.ROCKET_INDEX_TYPE) of HISTORY_REC_TYPE;

  POS_HISTORY       : HISTORY_LIST_TYPE;   -- :olds old rocket/target positions

  MOVE_NUMBER       : Types.WORD_INDEX;         -- to update display

  NEXT_ROCKET_MSG   : ROCKET_MSG_TYPE;          -- local copy of input msg
  NEXT_TARGET_LIST  : Target.TARGET_DATA_LIST_TYPE; -- local copy of input data
  GUIDE_MSG         : ROCKET_GUIDE_MSG_TYPE;    -- local copy of output msg

  AIMPOINT_LIST     : AIMPOINT_LIST_TYPE(Types.ROCKET_INDEX_TYPE);
                                                -- local copy
  MOVE_ROCKETS      : Graphics.MOVE_LIST_TYPE(Types.ROCKET_INDEX_TYPE);
  MOVE_INDEX        : Types.WORD_INDEX;

  PIXEL_POINT       : Shapes.PIXEL;
  MSG_INDEX         : Types.WORD_INDEX;         -- used to index incoming report
  OLD_SEQ_TAG       : Sync.SEQ_TYPE;            -- to filter stale reports out
  ANY_ACTIVE_ROCKETS : BOOLEAN;                 -- used to update OLD_SEQ_TAG
  ACTIVE_ROCKETS_ID  : Types.ROCKET_INDEX_TYPE; -- holds an active rockets ID

  NEXT_ENGAGED      : Target.TARGET_ID_TYPE;
  NEXT_DISENGAGED   : Target.TARGET_ID_TYPE;  -- keep track of all disengagements

  DISENGAGED_LIST   : array(Types.ROCKET_INDEX_TYPE) of Target.TARGET_ID_TYPE;
```

```
  DISENGAGED_ON_PTR  : Types.WORD_INDEX;
  DISENGAGED_OFF_PTR : Types.WORD_INDEX;
  DISENGAGED_ACK_PTR : Types.WORD_INDEX;

  AVAILABLE_ROCKET : Types.WORD_INDEX;          -- possible rocket to launch

  LAUNCH_PENDING   : BOOLEAN := FALSE;
  LAUNCH_TARGET    : Target.TARGET_ID_TYPE;
  LAUNCH_ROCKET    : Types.ROCKET_INDEX_TYPE;

  ROCKET_DESTROYED : BOOLEAN;
  ROCKET_LAUNCHED  : BOOLEAN;

begin
  accept Start;
  for I in AIMPOINT_INFO'range loop      -- initialize track data
    AIMPOINT_INFO(I).ACTIVE := FALSE;
    DISENGAGED_LIST(I) := 0;
  end loop;
  NEXT_ENGAGED := 0;

  DISENGAGED_ON_PTR  := 1;               -- initialize disengage circle queue
  DISENGAGED_OFF_PTR := 1;
  DISENGAGED_ACK_PTR := 1;

  OLD_SEQ_TAG := 0;

  loop                                   -- Main processing loop
    begin                                  -- exception block
      Time_Stamp.Log(0002);      --$TP(0002) Control task start time
      ROCKET_DESTROYED := FALSE;
      ROCKET_LAUNCHED  := FALSE;
      ANY_ACTIVE_ROCKETS := FALSE;
--
-- Rendezvous with buffer task to get next rocket message from sensor
--
      Time_Stamp.Log(0003); --$TP(0003) Control rendezvous with Report_Buf start
      RDL.Report_Buf.Get_Report(NEXT_ROCKET_MSG);
      Time_Stamp.Log(0004); --$TP(0004) Control rendezvous with Report_Buf end
--
--  If there are more on circular disengage queue, send another to tracker
--
      if DISENGAGED_OFF_PTR /= DISENGAGED_ON_PTR and then
        not NEXT_TARGET_LIST(DISENGAGED_LIST(DISENGAGED_OFF_PTR)).STATUS.ENGAGED
      then
        DISENGAGED_OFF_PTR := DISENGAGED_OFF_PTR rem dis_list_size + 1;
      end if;
      if DISENGAGED_OFF_PTR = DISENGAGED_ON_PTR then
        NEXT_DISENGAGED := 0;
      else
        NEXT_DISENGAGED := DISENGAGED_LIST(DISENGAGED_OFF_PTR);
```

```
      end if;
--
--  Rendezvous to Get target list from target tracker, and provide it
--  with information on which targets have been engaged and disengaged.
--
      Time_Stamp.Log(0005);  --$TP(0005) Control rendezvous with Track_Dat start
      Target.Track_Data.Get(NEXT_TARGET_LIST, NEXT_ENGAGED, NEXT_DISENGAGED);
      Time_Stamp.Log(0006);  --$TP(0006) Control rendezvous with Track_Dat end
--
--  Check if Track task has recognized the engage request, if so then
--  it is safe to clear it, and possibly engage another.
--
      if NEXT_ENGAGED /= 0 and then
         NEXT_TARGET_LIST(NEXT_ENGAGED).STATUS.ENGAGED
      then
        NEXT_ENGAGED := 0;
      end if;
--
--  Check to see if last disengage request was acknowledged
--
      if DISENGAGED_ACK_PTR /= DISENGAGED_OFF_PTR and then
        not NEXT_TARGET_LIST(DISENGAGED_LIST(DISENGAGED_ACK_PTR)).STATUS.ENGAGED
      then
        DISENGAGED_ACK_PTR := DISENGAGED_ACK_PTR rem dis_list_size + 1;
      end if;

--
-- determine which rockets have been expended, and delete them from screen
-- (previously active, but no longer in report list)
--
      MOVE_INDEX := 0;
      MSG_INDEX := 1;
      for ROCKET_ID in Types.ROCKET_INDEX_TYPE loop

        if AIMPOINT_INFO(ROCKET_ID).ACTIVE then
          if NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).SYNC_TAG = OLD_SEQ_TAG then
            ANY_ACTIVE_ROCKETS := TRUE;  -- need an active rockets time tag
            ACTIVE_ROCKETS_ID := ROCKET_ID;
            exit;                              -- old rocket report
          end if;

--
--  look at most recent rocket report message to make sure rocket is still alive
--

          if MSG_INDEX <= NEXT_ROCKET_MSG.NUM_ROCKETS and then
             ROCKET_ID = NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).ROCKET_ID
          then
            POS_HISTORY(ROCKET_ID).ROCKET_OLD :=
                             AIMPOINT_INFO(ROCKET_ID).ROCKET_POS;
            AIMPOINT_INFO(ROCKET_ID).ROCKET_POS :=
```

```
                        NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).POSITION;
          POS_HISTORY(ROCKET_ID).TARGET_OLD :=
                        AIMPOINT_INFO(ROCKET_ID).TARGET_POS;
          AIMPOINT_INFO(ROCKET_ID).TARGET_POS :=
        NEXT_TARGET_LIST(POS_HISTORY(ROCKET_ID).TARGET_AIMED_AT).POSITION_NEW;
          MOVE_INDEX := MOVE_INDEX + 1;
          MOVE_ROCKETS(MOVE_INDEX) :=
                (XY_OLD => Grid_to_Pixel(POS_HISTORY(ROCKET_ID).ROCKET_OLD),
                 XY_NEW => Grid_to_Pixel(AIMPOINT_INFO(ROCKET_ID).ROCKET_POS),
                 OBJECT => Shapes.ROCKET,
                 COLOR  => Graphics.ROCKET_COLOR);
          MSG_INDEX := MSG_INDEX + 1;
        else
--
--  the rocket has deceased, put it in the list for erasure.
--
          PIXEL_POINT := Grid_to_Pixel(      -- get last point in pixel value
                      AIMPOINT_INFO(ROCKET_ID).ROCKET_POS);
          AIMPOINT_INFO(ROCKET_ID).ACTIVE := FALSE; -- mark as inactive
          MOVE_INDEX := MOVE_INDEX + 1;
          MOVE_ROCKETS(MOVE_INDEX) :=
            (PIXEL_POINT,
             PIXEL_POINT,
             Shapes.ROCKET,
             Graphics.background_color);
          AVAILABLE_ROCKET := ROCKET_ID;        -- save if decide to launch
          DISENGAGED_LIST(DISENGAGED_ON_PTR):=
                       POS_HISTORY(ROCKET_ID).TARGET_AIMED_AT;
          DISENGAGED_ON_PTR  := DISENGAGED_ON_PTR rem dis_list_size + 1;
          Interrupt_Control.Disable;
          Status.STATUS_CONTROL(Status.AIRBORNE).DATA :=
             Status.STATUS_CONTROL(Status.AIRBORNE).DATA - 1;
          Status.STATUS_CONTROL(Status.EXPENDED).DATA :=
             Status.STATUS_CONTROL(Status.EXPENDED).DATA + 1;
          Interrupt_Control.Enable;
          ROCKET_DESTROYED := TRUE;
        end if;   -- found
      else
--
--    rocket slot previously inactive, see if rocket has launched
--
        if MSG_INDEX <= NEXT_ROCKET_MSG.NUM_ROCKETS and then
           NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).ROCKET_ID =
                                                    ROCKET_ID
        then
--
--    ROCKET HAS BEEN LAUNCHED, UPDATE DATA BASES
--
          AIMPOINT_INFO(ROCKET_ID) :=
             ( TRUE,                          -- ACTIVE
               NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).POSITION, -- NEW
```

```
              NEXT_TARGET_LIST(LAUNCH_TARGET).POSITION_NEW);   -- NEW
         POS_HISTORY(ROCKET_ID) :=
              (NEXT_ROCKET_MSG.ROCKET_LIST(MSG_INDEX).POSITION, -- OLD
               NEXT_TARGET_LIST(LAUNCH_TARGET).POSITION_NEW,    -- OLD
               LAUNCH_TARGET);              -- TARGET AIMED AT
         LAUNCH_PENDING := FALSE;                -- all accounted for
         MSG_INDEX := MSG_INDEX + 1;
         Interrupt_Control.Disable;
         Status.STATUS_CONTROL(Status.AIRBORNE).DATA :=
            Status.STATUS_CONTROL(Status.AIRBORNE).DATA + 1;
         Interrupt_Control.Enable;
         ROCKET_LAUNCHED := TRUE;
        else
         AVAILABLE_ROCKET := ROCKET_ID;
        end if;   -- new rocket test
      end if;     -- active test
    end loop;     -- rocket-id loop (scan of all rockets)
--
-- Update Time tag for next message.
--
    if ANY_ACTIVE_ROCKETS then
      OLD_SEQ_TAG := NEXT_ROCKET_MSG.ROCKET_LIST(ACTIVE_ROCKETS_ID).SYNC_TAG;
    end if;              -- if no active rockets, don't change OLD_SEQ_TAG.


--
-- Get guidance task(s) working on finding new aimpoint for guidance msg
--
    for I in Types.WORD_INDEX range 1..Distrib.num_guide_tasks loop
      Time_Stamp.Log(0007);     --$TP(0007) Control rendezvous with Guidance(1) start
      Rocket_Guide(I).History(
         AIMPOINT_INFO(Distrib.guide_low(I)..Distrib.guide_high(I)));
      Time_Stamp.Log(0008);     --$TP(0008) Control rendezvous with Guidance(1) end
    end loop;
--
-- update status information
--
    Interrupt_Control.Disable;
    if ROCKET_LAUNCHED then
      Status.STATUS_CONTROL(Status.AIRBORNE).DISPLAYED := FALSE;
    end if;
    if ROCKET_LAUNCHED or ROCKET_DESTROYED then
      Status.STATUS_CONTROL(Status.AIRBORNE).DISPLAYED := FALSE;
      Status.STATUS_CONTROL(Status.EXPENDED).DISPLAYED := FALSE;
      Status.REQ_COUNT := Status.REQ_COUNT + 1;
      if Status.REQ_COUNT = 1 then
        Time_Stamp.Log(0009);     --$TP(0009) Control rendezvous with Status start
        Status.Update.Signal;
        Time_Stamp.Log(0010);     --$TP(0010) Control rendezvous with Status end
      end if;
    end if;
    Interrupt_Control.Enable;
```

```
    MSG_INDEX := 0;              -- zero index for creating guidance message
--
--  Now, check if we should try to create a new ROCKET.  Note that
--  if a rocket has just been destroyed, don't try to fire a new one
--  before the rocket tracker knows that it has been disengaged.  Otherwise
--  it is likely to choose a target other than one that is closest.
--
    if not LAUNCH_PENDING and
       DISENGAGED_ACK_PTR   DISENGAGED_ON_PTR and    -- all have been ack'ed
       NEXT_ENGAGED = 0                              -- engage has been ack'ed
    then
      NEXT_ENGAGED := Engage(NEXT_TARGET_LIST);
      if NEXT_ENGAGED > 0 then
        LAUNCH_ROCKET := AVAILABLE_ROCKET;
        LAUNCH_TARGET := NEXT_ENGAGED;
        LAUNCH_PENDING := TRUE;
      end if;  -- ready to launch
    end if;   -- not pending check
--
-- get graphics task working on displaying rockets
--
    Time_Stamp.Log(0011);      --$TP(0011) Control rendezvous with Graphics start
    Graphics.Display.Move(Graphics.LOW, MOVE_ROCKETS(1..MOVE_INDEX));
    Time_Stamp.Log(0012);      --$TP(0012) Control rendezvous with Graphics end
--
-- now get results of guidance information
--
    for I in Types.WORD_INDEX range 1..Distrib.num_guide_tasks loop
      Time_Stamp.Log(0013);      --$TP(0013) Control rendezvous with Guidance(2) start
      Rocket_Guide(I).Next_Guidance(
        AIMPOINT_LIST(Distrib.guide_low(I)..Distrib.guide_high(I)) );
      Time_Stamp.Log(0014);      --$TP(0014) Control rendezvous with Guidance(2) end
    end loop;

--
--  Now generate new guidance message and send to Guide_Buf
--
--
    for ROCKET_ID in AIMPOINT_INFO'range loop
      if AIMPOINT_INFO(ROCKET_ID).ACTIVE then
        MSG_INDEX := MSG_INDEX + 1;
        GUIDE_MSG.ROCKET_GUIDE_LIST(MSG_INDEX) :=
                                  (ROCKET_ID,AIMPOINT_LIST(ROCKET_ID));
      elsif LAUNCH_PENDING and then
            ROCKET_ID = LAUNCH_ROCKET then
        MSG_INDEX := MSG_INDEX + 1;
-- initiate launch
        GUIDE_MSG.ROCKET_GUIDE_LIST(MSG_INDEX) := (ROCKET_ID,
                                (Config.launch_azimuth,
                                 Config.launch_elevation));
```

```
        end if;
      end loop;
      GUIDE_MSG.NUM_ROCKETS := MSG_INDEX;
      Time_Stamp.Log(0015);      --$TP(0015) Control rendezvous with Guide_Buf start
      RDL.Guide_Buf.Put_Guide(GUIDE_MSG);   -- send new guidance message
      Time_Stamp.Log(0016);      --$TP(0016) Control rendezvous with Guide_Buf end

    exception
      when others =>
        Debug_IO.Put_Line("Exception in Control task");
    end;                              -- exception block
  end loop;                           -- main processing loop
end Control_Type;   -- Rocket.Control task body
```

```
--------------------------------------------------------------------
--% UNIT:      Debug_IO Spec.                                       --
--% Effects:  Provides non-intrusive trace output to secondary port.   --
--% Modifies: No global data is modified.                           --
--% Requires: No initialization is required.                        --
--% Raises:   No explicitly raised exceptions are propagated.       --
--% Engineer: T. Griest.                                            --
--------------------------------------------------------------------

--|
--| PACKAGE SPEC : Debug_IO
--|
--|   This package is used to provide visibility to the character (and string)
--| input and output procedures.  Currently, because the screen memory is
--| written directly to, all text input and output is done via a serial port.
--| These routines are needed to signal to the user (via the serial port) that
--| an exception has occurred.
--|

-- Modifications Log
--
-- 88-09-01 : TEG => Original created.
--

package Debug_IO is

  procedure Put(CHAR : CHARACTER);
  procedure Get(CHAR : out CHARACTER);
  procedure Put(STR : STRING);
  procedure Get(STR : out STRING);
  procedure Put_Line(STR : STRING);
  procedure Get_Line(STR : out STRING; LENGTH : out INTEGER);
  procedure Skip_Line;

end Debug_IO;
```

```
------------------------------------------------------------------------
--% UNIT:      Debug_IO body.                                          --
--% Effects:  Provides non-intrusive trace output to secondary port.   --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
------------------------------------------------------------------------

--|
--| PACKAGE BODY : Debug_IO
--|
--|   The Debug_IO package is used to provide a means of communication from
--| the BDS to the user.  Since the terminal (the EGA screen in this case) is
--| being written to directly, output cannot take place there, and therefore
--| Text_IO cannot be used.  See the hardware configuration file for more
--| details on the input and output modes.
--|

-- Modifications Log
--
-- 88-09-01 : TEG => Original created.
--

with Terminal_Driver;
  pragma ELABORATE(Terminal_Driver);

package body Debug_IO is

pragma SUPPRESS(storage_check);

  procedure Put(CHAR : CHARACTER) is
  begin
    Terminal_Driver.Put_Character(CHAR);
  end Put;  -- character

  procedure Get(CHAR : out CHARACTER) is
  begin
    Terminal_Driver.Get_Character(CHAR);
  end Get;  -- character

  procedure Put(STR : STRING) is
  begin
    for I in STR'range loop
      Terminal_Driver.Put_Character(STR(I));
    end loop;
  end Put;    -- String

  procedure Get(STR : out STRING) is
  begin
    for I in STR'range loop
```

```
      Terminal_Driver.Get_Character(STR(I));
    end loop;
  end Get;  -- String

  procedure Put_Line(STR : STRING) is
  begin
    for I in STR'range loop
      Terminal_Driver.Put_Character(STR(I));
    end loop;
    Terminal_Driver.Put_Character(ASCII.CR);
    Terminal_Driver.Put_Character(ASCII.LF);
  end Put_Line;

  procedure Get_Line(STR : out STRING; LENGTH : out INTEGER) is
    CHAR : CHARACTER := ASCII.NUL;
    LEN  : INTEGER   := STR'first;
  begin
    while CHAR /= ASCII.CR and LEN <= STR'last loop
      Terminal_Driver.Get_Character(CHAR);
      STR(LEN) := CHAR;
      LEN := LEN + 1;
    end loop;
  end Get_Line;

  procedure Skip_Line is
    CHAR : CHARACTER := ASCII.NUL;
  begin
    while CHAR /= ASCII.CR loop
      Terminal_Driver.Get_Character(CHAR);
    end loop;
  end Skip_Line;


end Debug_IO;
```

```
-----------------------------------------------------------------
--% UNIT:      Distrib Package Spec.                            --
--% Effects:  Provides parameters to control task arrays and work lists.--
--% Modifies: No global data is modified other than in this spec.      --
--% Requires: Depends on presence of Distrbuted Runtime for # of tasks. --
--% Raises:   No explicitly raised exceptions are propagated.         --
--% Engineer: T. Griest.                                              --
-----------------------------------------------------------------

--|
--| PACKAGE SPEC : Distrib
--|
--| OPERATION :
--|   This package controls the parameters for automatically performing a
--| division of the guidance workload.  In this case, a large array can be
--| broken down so that two or more tasks can perform their operations on the
--| array at the same time (if true multi-processing is in effect).
--|

-- Modifications Log
--
-- 88-12-05 : TEG => Original Created.
-- 89-12-06 : TEG => Enhanced to support dynamic configuration/reconfiguration
--

with Types;
----------------------------------------------------
--  DISTRIBUTION CONTROL PARAMETERS              --
----------------------------------------------------
package Distrib is
--
--  Configuration Setting for number of Rockets and Targets
--  These are set during package body elaboration.
--
NUM_TARGETS  : Types.WORD_INDEX;
NUM_ROCKETS  : Types.WORD_INDEX;

--
--  Max_num_guide_tasks is used to determine the maximum number of guide
--  tasks which could be created.  It is used simply to define the size of
--  the index arrays.
--
  Max_guide_tasks   : constant := 2;
--
--  NUM_GUIDE_TASKS contains the ACTUAL number of guide tasks in the current
--  configuration.  It is initialized by a call to the distributed runtime
--  during package elaboration.
--
  NUM_GUIDE_TASKS       : Types.WORD_INDEX;

--
```

```
--  MASTER is TRUE iff this processor has been configured as the master
--  processor.
--
  MASTER               : BOOLEAN;
--
--  The following two "index" arrays are used by the Congrol task to
--  divide work among the possible guidance tasks.  These values are
--  also intialized according to the configuration control tables in
--  in the Distrib package body during elaboration.
--
  GUIDE_LOW        : array(Types.WORD_INDEX range 1..Max_guide_tasks)
                       of Types.WORD_INDEX;
  GUIDE_HIGH       : array(Types.WORD_INDEX range 1..Max_guide_tasks)
                       of Types.WORD_INDEX;
--
--  RESTART is used to stop operation of the BDS and allow the operator
--  setup a different configuration.  It is only called when the MODE
--  button is pressed while the RESET button is held down on the mouse.
--
  procedure Restart;    -- DOES NOT RETURN TO CALLER!
  pragma INTERFACE(ASM86, Restart);
  pragma INTERFACE_SPELLING(Restart, "D1DRTE?RESTART");

end Distrib;
```

```
-----------------------------------------------------------------------
--% UNIT:     Engage Procedure Spec.                                 --
--% Effects:  Determines if Rocket is to be launched, and at what target.-
--% Modifies: No global data is modified.                            --
--% Requires: Status package must set mode and airborne counts.      --
--% Raises:   No explicitly raised exceptions are propagated.        --
--% Engineer: M. Sperry.                                             --
-----------------------------------------------------------------------

--|
--| SUBPROGRAM SPEC : Engage
--|
--|   This function determines which target will be selected when it is
--| determined that a rocket needs a target to aim at.
--|

-- Modifications Log
--
-- 88-11-10 : MPS => Original Created.
--

with Target;

function Engage(TARGET_INFO : in  Target.TARGET_DATA_LIST_TYPE) return
               Target.TARGET_ID_TYPE;
```

```
------------------------------------------------------------------
--% UNIT:     Engage Procedure Body.                            --
--% Effects:  Determines if Rocket is to be launched, and at what target.-
--% Modifies: No global data is modified.                       --
--% Requires: Status package must set mode and airborne counts. --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: M. Sperry.                                        --
------------------------------------------------------------------

--|
--| SUBROGRAM BODY : Engage
--|
--|   The Engage procedures performs two functions based on the MODE.  The
--| MODE is either MANUAL or AUTOMATIC.  In MANUAL mode the engage procedure
--| first determines if a rocket can be launched and not exceed the maximum
--| allowable rockets.  It then reads the shared variables of the reticle's
--| position and the LAUNCH button on the mouse and determines if the reticle
--| is in proximity to a target.  If so, that target is chosen unless there is
--| one closer.  In AUTOMATIC mode, if there are not too many active rockets,
--| then the target closest to the bottom of the screen is chosen.  This routine
--| is called during every rocket control task iteration.  The returned
--| parameter TARGET is zero if no target should be engaged, otherwise it
--| indicates the selected targets id.
--|

-- Modifications Log
--
-- 88-11-20 : MPS => Original Created.
--

with Interrupt_Control;
with Status;
with Mouse_Buffer;
with Types;
with Config;
with Shapes;
with Time_Stamp;
with Distrib;
  pragma ELABORATE(Interrupt_Control, Status, Mouse_Buffer, Distrib);

function Engage(TARGET_INFO : in Target.TARGET_DATA_LIST_TYPE) return
                Target.TARGET_ID_TYPE is

use Types;      -- for operators
use Status;     -- for operators

RETICLE_X_PIXEL : Types.WORD;              -- reticle in PIXEL coordinates
RETICLE_Y_PIXEL : Types.WORD;              -- reticle in PIXEL coordinates
RETICLE_X_GRID  : Types.METERS;            -- reticle in GRID coordinates
RETICLE_Y_GRID  : Types.METERS;            -- reticle in GRID coordinates
PREV_DISTANCE   : Types.METERS;
```

```
DISTANCE_X      : Types.METERS;
DISTANCE_Y      : Types.METERS := Config.meters_in_battle_area;
TOTAL_DISTANCE  : Types.METERS;
TARGET_ID       : Target.TARGET_ID_TYPE;

begin
  Time_Stamp.Log(0018);     --$TP(0018) Engage start
  TARGET_ID := 0;                              -- default
  if Status.STATUS_CONTROL(Status.AIRBORNE).DATA <
     Types.WORD(Distrib.NUM_ROCKETS)
  then
    if Status.MODE = Status.MANUAL then
      if Mouse_Buffer.LAUNCH then
        -- read ABS_X and ABS_Y in Mouse_Buffer, then convert to METERS types.
        -- Then, find closest target in list to reticle, and give it back.
        Interrupt_Control.Disable;          -- go atomic while reading
        RETICLE_X_PIXEL := Mouse_Buffer.NEW_ABS_X;
        RETICLE_Y_PIXEL := Mouse_Buffer.NEW_ABS_Y;
        Mouse_Buffer.LAUNCH := FALSE;
        Interrupt_Control.Enable;
        RETICLE_X_GRID :=
        Types.METERS(Types.METERS(RETICLE_X_PIXEL -
                     Config.battlefield_screen_left) *
                     Types.METERS(Config.meters_per_X_pixel));
        RETICLE_Y_GRID :=
        Types.METERS(Types.METERS(Config.battlefield_screen_bottom  -
                     RETICLE_Y_PIXEL) *
                     Types.METERS(Config.meters_per_Y_pixel));
        -- This loop locates the closest target to the reticle center
        for ID in Types.TARGET_INDEX_TYPE loop
          if TARGET_INFO(ID).STATUS.ACTIVE and then
             not TARGET_INFO(ID).STATUS.ENGAGED then
            DISTANCE_X := abs(RETICLE_X_GRID - Types.METERS(
                          TARGET_INFO(ID).POSITION_NEW.X));
            DISTANCE_Y := abs(RETICLE_Y_GRID - Types.METERS(
                          TARGET_INFO(ID).POSITION_NEW.Y));
            if DISTANCE_X <= Shapes.reticle_X_error and
               DISTANCE_Y <= Shapes.reticle_Y_error
            then
              TOTAL_DISTANCE := Types.METERS(DISTANCE_X * DISTANCE_X) +
                                Types.METERS(DISTANCE_Y * DISTANCE_Y);
              if TARGET_ID = 0 or else TOTAL_DISTANCE < PREV_DISTANCE then
                PREV_DISTANCE := TOTAL_DISTANCE;
                TARGET_ID := ID;
              end if;                       -- distance/target check
            end if;                         -- x and y reticle distance check
          end if;                           -- active/not engaged check
        end loop;
      end if;                          -- launch check
    else                        -- automatic mode, search for closest Y value
      for ID in Types.TARGET_INDEX_TYPE loop
```

```
      if TARGET_INFO(ID).STATUS.ACTIVE and then
         (not TARGET_INFO(ID).STATUS.ENGAGED and
         Types.METERS(TARGET_INFO(ID).POSITION_NEW.Y) <= DISTANCE_Y)
      then
        DISTANCE_Y := Types.METERS(TARGET_INFO(ID).POSITION_NEW.Y);
        TARGET_ID := ID;
      end if;                   -- active/not engaged/closest y check
    end loop;
  end if;                       -- mode check
 end if;                        -- number of rockets check
 Time_Stamp.Log(0019);        --$TP(0019) Engage end
 return TARGET_ID;
end Engage;
```

```
-------------------------------------------------------------------
--% UNIT:     Graphics Package Spec.                              --
--% Effects:  Performs all updates to graphics display.           --
--% Modifies: No global data is modified.                         --
--% Requires: Screen must be put in graphics mode by runtime initialize.--
--% Raises:   QUEUE_ERROR is raised if no room for move list.     --
--% Engineer: T. Griest / M. Sperry.                              --
-------------------------------------------------------------------

--|
--| PACKAGE SPEC : Graphics
--|
--|   The Graphics package provides the interface for all screen display
--| operations.  All activity is performed by the Display task which insures
--| that the display is updated in a consistent and timely fashion.  The shapes
--| that the graphics displays are all defined in the Shapes package.  The
--| MOVE_RECORD is defined as all the elements needed in order to perform a
--| draw or erase of an image.  A MOVE_LIST is an array of MOVE_RECORDs and
--| it is used as a parameter when one of the routines responsible for moving
--| an image across the screen needs to rendezvous with Graphics.Display.  An
--| entire list is enqueued onto one of the priority queues and each element
--| is dequeued separately in order to continuously check for more arrivals.
--| The high priority queue is currently reserved for the reticle motion.
--|

-- Modifications Log
--
-- 88-08-25 : TEG => Original created.
--

with Types;
with Config;
with Shapes;

package Graphics is

stack_size       : constant := 8192;             -- in bytes

--
-- define screen and graphics constants
--

subtype COLOR_TYPE is Types.WORD; -- range 0..63;  -- 64 colors on EGA

background_color : constant COLOR_TYPE :=  0;     -- black
reticle_color    : constant COLOR_TYPE := 14;     -- bright yellow
border_color     : constant COLOR_TYPE :=  9;     -- bright blue
status_color     : constant COLOR_TYPE := 15;     -- bright white
status_box_color : constant COLOR_TYPE :=  9;     -- bright blue
rocket_color     : constant COLOR_TYPE := 12;     -- bright red
target_color     : constant array(Types.TARGET_CLASS_TYPE, BOOLEAN) of
```

```
        COLOR_TYPE := ((6, 14), (3, 11), (2, 10), (5, 13));
        -- different color for engage = false/true and target type
no_process        : constant COLOR_TYPE := 16;     -- don't process object color

--
-- define graphics data structures
--

type MOVE_RECORD is record
  XY_OLD          : Shapes.PIXEL;        -- previous position object held
  XY_NEW          : Shapes.PIXEL;        -- new position
  OBJECT          : Shapes.SYMBOL_TYPE;  -- list of relative offsets
  COLOR           : COLOR_TYPE;          -- color for that object
end record;

type MOVE_LIST_TYPE is array (Types.WORD_INDEX range <>) of MOVE_RECORD;
type PRIORITY_TYPE is (HIGH, LOW);

QUEUE_ERROR       : exception;                    if queue over/underflow


task type Display_Type is
  entry Print_Titles(X,Y   : Types.WORD;
                     TITLE : STRING;
                     COLOR : COLOR_TYPE);
  entry Move(PRIORITY : PRIORITY_TYPE; WORK_LIST : MOVE_LIST_TYPE);
  pragma PRIORITY(Config.display_priority);
end Display_Type;
for Display_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                                                stack_size);
Display           : Display_Type;

end Graphics;
```

```
--------------------------------------------------------------------------
--% UNIT:      Graphics Package Body                                    --
--% Effects:  Performs all updates to graphics display.               --
--% Modifies: No global data is modified.                              --
--% Requires: A method of access to the EGA BIOS calls.                --
--% Raises:   QUEUE_ERROR is raised if no room for move list.          --
--% Engineer: T. Griest / M. Sperry.                                   --
--------------------------------------------------------------------------

--|
--| PACKAGE BODY : Graphics
--|
--|   The purpose of the graphics package body is the implementation of the
--| display task.
--|
--|
--| TASK BODY : Graphics.Display
--|
--|   The display task is responsible for buffering the various tasks that want
--| to draw their particular symbol on the screen.  The task begins by placing
--| the screen (via BIOS calls) into high resolution mode 10h.  When this
--| is done, the screen will be in write mode 0 - the BIOS default.  In this
--| mode it is possible to print characters easily by calling the appropriate
--| BIOS routine.  After the statistics have been printed, a change to write
--| mode 2 is accomplished.  This mode permits quick drawing of pixels in the
--| color needed, and the battlefield border is drawn this way.  The rest of
--| the graphics are also done in this mode.  The display task then waits
--| for a work request to draw a symbol.  When a request comes in, it is put
--| on a prioritized queue.  The queue used is a function of the callers'
--| priority.  Now, since there is work to do, the task processes one symbol
--| at a time, checks to see if other tasks are waiting to queue any requests,
--| and continues processing until no requests are left in any of the queues.
--| When a request is processed, it's old position is erased, and it's new
--| position is drawn.  No attempt is made to synchronize with the vertical
--| retrace since it would slow down the task too much.  The penalty associated
--| with this is a slight flicker of some of the images (especially when the
--| reticle is being slowly dragged across the screen).  When checking if there
--| is more work to do, using 'count instead of a select statement was used
--| because the code generated for 'count was significantly smaller.
--|

-- Modifications Log
--
-- 88-08-25 : MPS => Original Created
--

with Machine_Dependent;
with Interrupt_Control;
with Debug_IO;
with Time_Stamp;
pragma ELABORATE(Machine_Dependent, Interrupt_Control, Debug_IO, Time_Stamp);
```

```
package body Graphics is

task body Display_Type is
use Types;                              -- needed for visibility to "+" operator

buffer_size       : constant := 256;
initialize_screen : constant := 0;      -- for Int 10 BIOS call, fnctn 0
dummy_1           : constant := 0;      -- dummy parameter
dummy_2           : constant := 0;      -- dummy parameter
position_cursor   : constant := 2;      -- position function is int 10, fnct 2
write             : constant := 14;     -- write char is int 10, fnct 16#0E#

type CIRCULAR_BUFFER is array(Types.WORD_INDEX range 0 .. buffer_size - 1) of
                                                            MOVE_RECORD;

type BUFFER_TYPE is record
  ON   : Types.WORD_INDEX := 0;
  OFF  : Types.WORD_INDEX := 0;
  DATA : CIRCULAR_BUFFER;
end record;

SET_PRIORITY : PRIORITY_TYPE := PRIORITY_TYPE'FIRST;
BUFFER       : array(PRIORITY_TYPE'FIRST..PRIORITY_TYPE'LAST) of BUFFER_TYPE; -- set up queues
NO_WORK      : BOOLEAN;                 -- all queues empty?
WORK_REQUEST : MOVE_RECORD;             -- for individual processing
OBJECT       : Shapes.OBJECT_PTR;       -- current object to move
TEXT_MODE    : BOOLEAN;                 -- printing stats titles?
CHAR         : Types.WORD;              -- temp for holding string slices
COUNTER      : Types.WORD;              -- index into TITLE string


procedure Erase_Image(BASE : Shapes.PIXEL;
                      ITEM : Shapes.OBJECT_PTR) is

--|
--| SUBPROGRAM BODY : Graphics.Display.Erase_Image
--|
--|  A procedure designed to calculate absolute coordinates for the routine
--| Machine_Dependant.Put_Pixel given a shape(OBJECT_PTR) and an absolute
--| reference point where the object is to be placed.  No color is specified
--| because the intent of this procedure is to erase, which is actually
--| drawing over the old image in the background color.
--|
begin
  Time_Stamp.Log(0020);     --$TP(0020) Graphics.Erase_Image start
  for I in ITEM.all'range loop
    Machine_Dependent.Put_Pixel(BASE.X + ITEM.all(I).X_OFFSET,
                                BASE.Y + ITEM.all(I).Y_OFFSET,
                                background_color);
  end loop;
```

```
  Time_Stamp.Log(0021);      --$TP(0021) Graphics.Erase_Image end
end Erase_Image;
pragma INLINE(ERASE_IMAGE);


procedure Draw_Image(BASE  : Shapes.PIXEL;
                     ITEM  : Shapes.OBJECT_PTR;
                     COLOR : COLOR_TYPE) is

--|
--| SUBPROGRAM BODY : Graphics.Display.Draw_Image
--|
--|   This procedure is functionally the same as Erase_Image except that a
--| color is passed to it so that the object can be drawn in that color.
--|

begin
  Time_Stamp.Log(0022);      --$TP(0022) Graphics.Draw_Image start
  for I in ITEM.all'range loop
    Machine_Dependent.Put_Pixel(BASE.X + ITEM.all(I).X_OFFSET,
                                BASE.Y + ITEM.all(I).Y_OFFSET,
                                COLOR);
  end loop;
  Time_Stamp.Log(0023);      --$TP(0023) Graphics.Draw_Image end
end Draw_Image;
pragma INLINE(DRAW_IMAGE);


procedure Initialize_Border is

--|
--| SUBPROGRAM BODY : Graphics.Display.Initialize_Border
--|
--|   A procedure which utilizes the Shapes package to place a color border
--| around the screen thus defining the battlefield area.  The reticle never
--| leaves the battlefield area and statistics are never displayed inside
--| the battlefield area.
--|

BORDER : MOVE_RECORD;

begin
  BORDER.OBJECT := Shapes.DOT;
  OBJECT := Shapes.OBJECT_PTR_TABLE(BORDER.OBJECT);
  BORDER.COLOR := border_color;
--
-- draw top and bottom border
--
  for I in Config.border_left..Config.border_right loop
    BORDER.XY_NEW := (Types.COORDINATE(I),Config.border_top);
    Draw_Image(BORDER.XY_NEW,OBJECT,BORDER.COLOR);
```

```
    BORDER.XY_NEW := (Types.COORDINATE(I),Config.border_bottom);
    Draw_Image(BORDER.XY_NEW,OBJECT,BORDER.COLOR);
  end loop;
--
-- draw left side and right side border
--
  for J in Config.border_top..Config.border_bottom loop
    BORDEP.XY_NEW := (Config.border_left,Types.COORDINATE(J));
    Draw_Image(BORDER.XY_NEW,OBJECT,BORDER.COLOR);
    BORDER.XY_NEW := (Config.border_right,Types.COORDINATE(J));
    Draw_Image(BORDER.XY_NEW,OBJECT,BORDER.COLOR);
  end loop;
exception
  when others => Debug_IO.Put_Line("Exception raised in Graphics.Initialize");
end Initialize_Border;


procedure Enqueue(PRIORITY : PRIORITY_TYPE; MOVE_REQUEST : MOVE_RECORD) is

--|
--| SUBPROGRAM BODY : Graphics.Display.Enqueue
--|
--|   A procedure which enqueues a MOVE_RECORD (a record containing all the
--| information needed to draw a symbol) onto the proper priority queue for
--| later processing.  May raise QUEUE_ERROR.
--|

ON_NEW   : Types.WORD_INDEX;

begin
  Time_Stamp.Log(0024);      --$TP(0024) Graphics.Enqueue start
  ON_NEW := (BUFFER(PRIORITY).ON + 1) rem buffer_size;
  if ON_NEW = BUFFER(PRIORITY).OFF then
    raise QUEUE_ERROR;
  end if;
  Interrupt_Control.Disable;                    -- compiler bug
  BUFFER(PRIORITY).DATA(ON_NEW) := MOVE_REQUEST;
  Interrupt_Control.Enable;                     --
  BUFFER(PRIORITY).ON := ON_NEW;
  Time_Stamp.Log(0025);      --$TP(0025) Graphics.Enqueue end
end Enqueue;
pragma INLINE(Enqueue);


procedure Dequeue(PRIORITY : PRIORITY_TYPE; MOVE_REQUEST : out MOVE_RECORD) is

--|
--| SUBPROGRAM BODY : Graphics.Display.Dequeue
--|
--|   A procedure which is given the priority of the queue it needs to access
--| in order to pop the MOVE_RECORD (a record containing drawing information)
```

```
--| off that queue.  If there are no items on that queue, QUEUE_ERROR is raised.
--|

OFF_NEW : Types.WORD_INDEX;

begin
  Time_Stamp.Log(0026);      --$TP(0026) Graphics.Dequeue start
  if BUFFER(PRIORITY).OFF = BUFFER(PRIORITY).ON then
    raise QUEUE_ERROR;
  end if;
  OFF_NEW := (BUFFER(PRIORITY).OFF + 1) rem buffer_size;
  Interrupt_Control.Disable;                  -- compiler bug
  MOVE_REQUEST := BUFFER(PRIORITY).DATA(OFF_NEW);
  Interrupt_Control.Enable;                   --
  BUFFER(PRIORITY).OFF := OFF_NEW;
  Time_Stamp.Log(0027);      --$TP(0027) Graphics.Dequeue end
end Dequeue;
pragma INLINE(Dequeue);


------------------------------------
--  Body of DISPLAY TASK          --
------------------------------------
begin
  NO_WORK := TRUE;
  TEXT_MODE := TRUE;
  Machine_Dependent.Int10(initialize_screen,
                          dummy_1,            -- dummy variables are unused
                          dummy_2);           -- hi-res graphics mode
  Machine_Dependent.Write_Mode_0;
  while TEXT_MODE loop
    accept Print_Titles(X,Y   : Types.WORD;
                        TITLE : STRING;
                        COLOR : COLOR_TYPE) do
      if TITLE'length > 0 then
        Machine_Dependent.Int10(position_cursor,X,Y);
        COUNTER := 1;
        while COUNTER <= TITLE'length loop
          CHAR := Types.WORD(CHARACTER'pos(TITLE(INTEGER(COUNTER))));
          Machine_Dependent.Int10(write,
                                  CHAR,
                                  COLOR);
          COUNTER := COUNTER + 1;
        end loop;
      else
        TEXT_MODE := FALSE;
      end if;
    end Print_Titles;
  end loop;
  Machine_Dependent.Write_Mode_2;             -- go to write mode 2
  Initialize_Border;                          -- draw battlefield border
```

```
  loop
    begin                                           -- exception block
      Time_Stamp.Log(0028);     --$TP(0028) Graphics task start
      if NO_WORK or Move'COUNT > 0 then
        Time_Stamp.Log(0112);     --$TP(0112) Graphics accept Move start
        accept Move(PRIORITY : PRIORITY_TYPE; WORK_LIST : MOVE_LIST_TYPE) do
          for I in WORK_LIST'range loop
            Enqueue(PRIORITY, WORK_LIST(I));
          end loop;
        end Move;
        Time_Stamp.Log(0113);     --$TP(0113) Graphics accept Move end
        NO_WORK := FALSE;
      end if;
--
--  Now there is some work to do, see if any left on highest priority
--
      SET_PRIORITY := PRIORITY_TYPE'FIRST;
      loop
        if BUFFER(SET_PRIORITY).ON /= BUFFER(SET_PRIORITY).OFF then
          Dequeue(SET_PRIORITY,WORK_REQUEST); -- at this point, requests real
          OBJECT := Shapes.OBJECT_PTR_TABLE(WORK_REQUEST.OBJECT);
          Erase_Image(WORK_REQUEST.XY_OLD, OBJECT);
          Draw_Image (WORK_REQUEST.XY_NEW, OBJECT, WORK_REQUEST.COLOR);
          NO_WORK := FALSE;
          exit;                         -- leave loop if we processed a request
        else
          NO_WORK := TRUE;              -- default
          exit when SET_PRIORITY = PRIORITY_TYPE'LAST;
          SET_PRIORITY := PRIORITY_TYPE'SUCC(SET_PRIORITY);
        end if;
      end loop;

    exception
      when QUEUE_ERROR => null;         -- since error is propagated to caller
      when others =>
        Debug_IO.Put_Line("Error in Display Task");
    end;                                -- exception block
  Time_Stamp.Log(0029);     --$TP(0029) Graphics task end
  end loop;
end Display_Type;


end Graphics;
```

```
-------------------------------------------------------------------------
--% UNIT:     Grid_to_Pixel Function Spec.                             --
--% Effects:  Converts battlefield meters X-Y to graphics Pixel X-Y.   --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
-------------------------------------------------------------------------

--|
--| SUBPROGRAM SPEC : Grid_To_Pixel
--|
--|   This function provides a translation to go from the "real" battlefield
--| to the screen battlefield.  Note that the screen battlefield has the Y
--| component at 0 at the top of the screen and increasing positively down
--| the screen.  A diagram in hwconfig.as shows the complete screen.
--|

-- Modifications Log
--
-- 88-09-26 : TEG => Original created.
--

with Shapes;
with Types;

function Grid_to_Pixel(GRID : in Types.POSITION_TYPE) return Shapes.Pixel;
pragma INLINE(Grid_to_Pixel);
```

```
---------------------------------------------------------------------------
--% UNIT:     Grid_to_Pixel Function Spec.                                 --
--% Effects:  Converts battlefield meters X-Y to graphics Pixel X-Y.       --
--% Modifies: No global data is modified.                                  --
--% Requires: No initialization is required.                               --
--% Raises:   No explicitly raised exceptions are propagated.              --
--% Engineer: T. Griest.                                                   --
---------------------------------------------------------------------------

--|
--| SUBPROGRAM BODY : Grid_To_Pixel
--|
--|   Translate from Battlefield Grid coordinates in meters to pixels
--| on the screen.  This means applying scale factors for x/y and
--| providing offsets to battlefield area on screen.  NOTE: since
--| battlefield coordinates have 0,0 in lower left; and graphics
--| coordinates have 0,0 in upper left, this involves a transpose of
--| the Y axis (thus the '-').
--|

-- Modifications Log
--
-- 88-10-20 : TEG => Original created.
-- 89-01-04 : MPS => Changed Time_Stamp to properly time the routine.
--

with Config;
with Time_Stamp;
with Math;
  pragma ELABORATE(Time_Stamp, Math);

function Grid_to_Pixel(GRID : in Types.POSITION_TYPE) return Shapes.Pixel is
  use Types;
  use Math;
  TEMP : Types.LONG_FIXED;
  PIX  : Shapes.PIXEL;
begin
  Time_Stamp.Log(0030);      --$TP(0030) Grid_To_Pixel start
  TEMP := GRID.X / Types.LONG_FIXED(Config.meters_per_x_pixel);
  PIX.X := Config.battlefield_screen_left + Types.COORDINATE(TEMP);
  TEMP := GRID.Y / Types.LONG_FIXED(Config.meters_per_y_pixel);
  PIX.Y := Config.battlefield_screen_left + Types.COORDINATE(TEMP);
  Time_Stamp.Log(0031);      --$TP(0031) Grid_To_Pixel end
  return PIX;
end Grid_to_Pixel;
```

```
-----------------------------------------------------------------------------
--% UNIT:     Guidance Task Subunit                                        --
--% Effects:  Calls "Guide" to compute next rocket aimpoint for every      --
--%           active rocket in the input list.                             --
--% Modifies: No global data is modified.                                  --
--% Requires: No initialization is required.                               --
--% Raises:   No explicitly raised exceptions are propagated.              --
--% Engineer: T. Griest.                                                   --
-----------------------------------------------------------------------------

--|
--| TASK BODY : Rocket.Guidance
--|
--|   Task Guidance is used as a template for an array of tasks which compute
--| guidance information for a specified number of rockets.  The first thing
--| it does is it gets the history information for the rocket/target list
--| and makes a local copy.  The index of the history array (containing previous
--| positions and which rockets were previously active) is ROCKET_ID.  The
--| entire guide_list array is passed, even though many of the entries may be
--| inactive.  Only active rockets (those that are in the air or taking off)
--| are given guidance.  The entire array however is again passed back to the
--| caller, the rocket control task.
--|

-- Modifications Log
--
-- 88-10-12 : TEG => Original created.
-- 89-11-22 : MPS => Adjusted to work with new Guide procedure.
--

with Guide;
with Time_Stamp;
with Interrupt_Control;
  pragma ELABORATE(Guide, Time_Stamp, Interrupt_Control);

separate(Rocket)

task body Guidance_Type is

  use Types;              -- for operator visibility
  NEXT_GUIDE_LIST    : AIMPOINT_LIST_TYPE(1..Config.max_rockets);
  NEXT_HISTORY_LIST  : POSITION_LIST_TYPE(1..Config.max_rockets);
  FIRST_ROCKET_ID    : Types.WORD_INDEX;
  LAST_ROCKET_ID     : Types.WORD_INDEX;

begin
  loop                        -- main processing loop
    begin                         -- exception block
    Time_Stamp.Log(0032);     --$TP(0032) Guidance task start
      Time_Stamp.Log(0033);     --$TP(0033) Guidance accept History start
      accept History(AIM_DATA : in POSITION_LIST_TYPE) do
```

```
        FIRST_ROCKET_ID := AIM_DATA'first;
        LAST_ROCKET_ID  := AIM_DATA'last;
        Interrupt_Control.Disable;    --BUGFIX for compiler bug (direction flag)
        NEXT_HISTORY_LIST(FIRST_ROCKET_ID..LAST_ROCKET_ID) := AIM_DATA;
        Interrupt_Control.Enable;     --BUGFIX for compiler bug
      end History;
      Time_Stamp.Log(0034);     --$TP(0034) Guidance accept History end
--
--  process list to create guidance information
--
      for ROCKET_ID in FIRST_ROCKET_ID..LAST_ROCKET_ID loop
        if NEXT_HISTORY_LIST(ROCKET_ID).ACTIVE then
          Guide(ROCKET_ID,NEXT_HISTORY_LIST(ROCKET_ID).ROCKET_POS,
                NEXT_HISTORY_LIST(ROCKET_ID).TARGET_POS,
                NEXT_GUIDE_LIST(ROCKET_ID));
        end if;
      end loop;

      Time_Stamp.Log(0035);     --$TP(0035) Guidance accept Next_Guidance start
      accept Next_Guidance(AIMPOINT_LIST : out AIMPOINT_LIST_TYPE) do
        if AIMPOINT_LIST'first /= FIRST_ROCKET_ID or  -- make sure list hasn't
           AIMPOINT_LIST'last  /= LAST_ROCKET_ID      -- changed from History
        then
          raise GUIDANCE_LIST_ERROR;
        else
          Interrupt_Control.Disable;  --BUGFIX for compiler bug (direction flag)
          AIMPOINT_LIST := NEXT_GUIDE_LIST(FIRST_ROCKET_ID..LAST_ROCKET_ID);
          Interrupt_Control.Enable;   --BUGFIX for compiler bug (direction flag)
        end if;
      end Next_Guidance;
      Time_Stamp.Log(0036);     --$TP(0036) Guidance accept Next_Guidance end

    exception
      when others =>
        Debug_IO.Put_Line("Error in GUIDANCE TASK");
    end;                             -- exception block
  Time_Stamp.Log(0037);      --$TP(0037) Guidance task end
  end loop;                   -- main processing loop
end Guidance_Type;
```

```
------------------------------------------------------------------------
--% UNIT:     Guide Function Spec.                                     --
--% Effects:  Computes a new aimpoint based on rocket/target positions. --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
------------------------------------------------------------------------

--|
--| SUBPROGRAM SPEC : Guide
--|
--|  The Guide function is used to find an aimpoint for the rocket to fly at
--| when it is in flight.  This includes guidance for the rocket during it's
--| launch phase.  It takes as parameters the rocket_index, the latest positions
--| of both the rocket and target and returns two Binary Angle Measuements,
--| one Azimuth and one Elevation per call.
--|

-- Modifications Log
--
-- 88-12-05 : TEG => Original created.
-- 89-11-07 : TEG => Changed from a function to a procedure call.
--

with Types;

procedure Guide(ROCKET_ID    : Types.ROCKET_INDEX_TYPE;
                ROCKET_POS   : Types.POSITION_TYPE;
                TARGET_POS   : Types.POSITION_TYPE;
                NEW_AIMPOINT : out Types.AIMPOINT_TYPE);
--pragma INLINE(Guide);
```

```
----------------------------------------------------------------------
--% UNIT:      Guide Function Body.                                  --
--% Effects:  Computes a new aimpoint based on rocket/target positions. --
--% Modifies: Aim_Data Rocket Info is modified.                       --
--% Requires: No initialization is required.                         --
--% Raises:   No explicitly raised exceptions are propagated.         --
--% Engineer: T. Griest.                                              --
----------------------------------------------------------------------

--|
--| SUBPROGRAM BODY : Guide
--|
--|  The Guide function takes the most recent two postions of a rocket/target
--| pair, and computes an aimpoint for the rocket to intercept.
--|  Because the target is assumed to be moving, a process which
--| extrapolates the target's position forward is used.  However, this section
--| is only called upon when the rocket is close to the target (TIME_TO_TARGET).
--|  The basic theory of operation is to control the rocket attitude by changing
 -| the previous aimpoint incrementally according to the amount of change
--| desired in the acceleration from the last interval.
--|

-- Modifications Log
--
-- 88-11-09 : TEG => Original created.
-- 89-11-14 : TEG => Equations were improved upon to be more realistic.
--

with Config;
with Types;
with Math;
with Rocket;
with Aim_Data;
with Aimpoint;  -- function
  pragma ELABORATE(Math,Aimpoint);

procedure Guide(ROCKET_ID    : Types.ROCKET_INDEX_TYPE;
                ROCKET_POS   : Types.POSITION_TYPE;
                TARGET_POS   : Types.POSITION_TYPE;
                NEW_AIMPOINT : out Types.AIMPOINT_TYPE) is

  use Types;      -- for operators
  use Aim_Data;   -- for enumeration types AXIS (x,y,z)
  use Math;       -- for speedy fixed point math

  accuracy             : constant := 1.0; -- resolution on TIME_TO_TARGET
  height_factor        : constant := 6;   -- boost done when z >= 1/6 (dist x_y)
  integration_interval : constant := 4.0; -- periods to integration acceleration
  integration_int_sq   : constant := (integration_interval-1.0) ** 2;
  furthest_extrapolate : constant := 300.0; -- don't bother going beyond
  max_change           : constant := 3.0; -- maximum change to acceleration
```

```
  limit_rock_extrap    : constant Types.LONG_FIXED := Types.LONG_FIXED(
                   (Config.meters_in_battle_area + Types.LONG_FIXED(1000.0)));

  ROCKET_VELOC_1       : Aim_Data.RATE_REC_TYPE;
  ROCKET_VELOC_2       : Aim_Data.RATE_REC_TYPE;
  TARGET_VELOC_1       : Aim_Data.RATE_REC_TYPE;
  ROCKET_ACCEL         : Aim_Data.RATE_REC_TYPE;
  ROCK_TARG_DELTA      : Types.POSITION_TYPE;
  BOOST_LIMIT          : Types.LONG_FIXED;
  ROCK_TARG_DSQ_X      : Types.LONG_FIXED;
  ROCK_TARG_DSQ_Y      : Types.LONG_FIXED;
  ROCK_TARG_DSQ_Z      : Types.LONG_FIXED;
  ROCK_TARG_DIST       : Types.LONG_FIXED;
  ROCK_TARG_XY_DIST    : Types.LONG_FIXED;
  ROCK_SQ_X            : Types.RATE_TYPE;
  ROCK_SQ_Y            : Types.RATE_TYPE;
  ROCK_SQ_Z            : Types.RATE_TYPE;
  ROCK_VELOC_VECT      : Types.RATE_TYPE;
  ROCK_XY_VELOC_VECT   : Types.RATE_TYPE;
  TIME_TO_TARGET       : Types.LONG_FIXED;
  EXTRAP_TARG          : Types.POSITION_TYPE;
  EXTRAP_ROCK          : Types.POSITION_TYPE;
  DESIRED_VELOC        : Aim_Data.RATE_REC_TYPE;
  DESIRED_ACCEL        : Aim_Data.RATE_REC_TYPE;
  CHANGE_ACCEL         : Aim_Data.RATE_REC_TYPE;
  SUM                  : Types.LONG_FIXED;
  SUM_VELOCITY         : Types.LONG_FIXED;
  AZIMUTH              : Types.BAM;
  ELEVATION            : Types.BAM;
  INTEGRATION_PERIOD   : Types.LONG_FIXED;
  INTEGRATION_SQ       : Types.LONG_FIXED;
begin
--
-- If a new launch is taking place, initialize the Aim_Data data base.
--
  if ROCKET_POS.Y = Config.launch_y and ROCKET_POS.X = Config.launch_x then
    Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_TARG := TARGET_POS;
    Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_TARG := TARGET_POS;
    Aim_Data.ROCKET_INFO(ROCKET_ID).PREV_ROCK := ROCKET_POS;
    Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK := ROCKET_POS;
    Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_ROCK := ROCKET_POS;
    Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT :=
                          (Config.launch_elevation,Config.launch_azimuth);
    Aim_Data.ROCKET_INFO(ROCKET_ID).BOOST_PHASE := TRUE;
  end if;
--
--  First update history of data.
--
  Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_TARG :=
                             Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_TARG;
  Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_TARG := TARGET_POS;
```

```
  Aim_Data.ROCKET_INFO(ROCKET_ID).PREV_ROCK :=
                              Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK;
  Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK :=
                              Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_ROCK;
  Aim_Data.ROCKET_INFO(ROCKET_ID).CURR_ROCK := ROCKET_POS;
--
-- First check Target's Y coordinate to avoid friendly fire.
-- IF ROCKET IS GOING OVER TARGET, SIMPLY SET AIMPOINT STRAIGHT DOWN.
--
  if TARGET_POS.Y < ROCKET_POS.Y then
    Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT := (ELEVATION => -16384,
            AZIMUTH => Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT.AZIMUTH);
    NEW_AIMPOINT := Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT;
    return;
  end if;
--
--  Compute Rocket Velocity in all three axes.
--
  ROCKET_VELOC_1.X := Types.RATE_TYPE(ROCKET_POS.X -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.X); -- rocket change X
  ROCKET_VELOC_1.Y := Types.RATE_TYPE(ROCKET_POS.Y -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.Y); -- rocket change Y
  ROCKET_VELOC_1.Z := Types.RATE_TYPE(ROCKET_POS.Z -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.Z); -- rocket change Z

  ROCKET_VELOC_2.X := Types.RATE_TYPE(
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.X -
            Aim_Data.ROCKET_INFO(ROCKET_ID).PREV_ROCK.X); -- rocket change X
  ROCKET_VELOC_2.Y := Types.RATE_TYPE(
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.Y -
            Aim_Data.ROCKET_INFO(ROCKET_ID).PREV_ROCK.Y); -- rocket change Y
  ROCKET_VELOC_2.Z := Types.RATE_TYPE(
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_ROCK.Z -
            Aim_Data.ROCKET_INFO(ROCKET_ID).PREV_ROCK.Z); -- rocket change Z
--
--  Compute Target Velocity in all three axes.
--
  TARGET_VELOC_1.X := Types.RATE_TYPE(TARGET_POS.X -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_TARG.X); -- target change X
  TARGET_VELOC_1.Y := Types.RATE_TYPE(TARGET_POS.Y -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_TARG.Y); -- target change Y
  TARGET_VELOC_1.Z := Types.RATE_TYPE(TARGET_POS.Z -
            Aim_Data.ROCKET_INFO(ROCKET_ID).LAST_TARG.Z); -- target change Z
--
--  Compute Acceleration for Rocket in all three axes.
--
  ROCKET_ACCEL.X := ROCKET_VELOC_1.X - ROCKET_VELOC_2.X;
  ROCKET_ACCEL.Y := ROCKET_VELOC_1.Y - ROCKET_VELOC_2.Y;
  ROCKET_ACCEL.Z := ROCKET_VELOC_1.Z - ROCKET_VELOC_2.Z;
--
--  Compute velocity vector for rocket using the formula
```

```
--   v = sqrt(curr_rock.X**2 + curr_rock.y**2 + curr_rock.z**2)
--
  ROCK_SQ_X := ROCKET_VELOC_1.X * ROCKET_VELOC_1.X;
  ROCK_SQ_Y := ROCKET_VELOC_1.Y * ROCKET_VELOC_1.Y;
  ROCK_SQ_Z := ROCKET_VELOC_1.Z * ROCKET_VELOC_1.Z;

  SUM_VELOCITY := Types.LONG_FIXED(ROCK_SQ_X) + Types.LONG_FIXED(ROCK_SQ_Y);
  ROCK_XY_VELOC_VECT := Types.RATE_TYPE(Math.Sqrt(SUM_VELOCITY));
  ROCK_VELOC_VECT := Types.RATE_TYPE(Math.Sqrt(SUM_VELOCITY +
                     Types.LONG_FIXED(ROCK_SQ_Z)));
--
--  Compute distance between rocket and target using the formula
--      d = sqrt(d(X)**2 + d.Y**2 + d.Z**2)
--      where d(i) = curr_rock(i) - curr_targ(i)
--
  ROCK_TARG_DELTA.X :=  TARGET_POS.X - ROCKET_POS.X;
  ROCK_TARG_DELTA.Y :=  TARGET_POS.Y - ROCKET_POS.Y;
  ROCK_TARG_DELTA.Z :=  TARGET_POS.Z - ROCKET_POS.Z;

  ROCK_TARG_DSQ_X := ROCK_TARG_DELTA.X * ROCK_TARG_DELTA.X;
  ROCK_TARG_DSQ_Y := ROCK_TARG_DELTA.Y * ROCK_TARG_DELTA.Y;
  ROCK_TARG_DSQ_Z := ROCK_TARG_DELTA.Z * ROCK_TARG_DELTA.Z;

  SUM := ROCK_TARG_DSQ_X + ROCK_TARG_DSQ_Y + ROCK_TARG_DSQ_Z;
  ROCK_TARG_DIST := Math.Sqrt(SUM);
--
--  Compute rocket time to target, ITERATION TAKES INTO ACCOUNT
--  changes in rocket velocity and target motion (NOTE: change in
--  rocket acceleration is NOT included)
--
  if ROCK_VELOC_VECT > 0.01 then
    TIME_TO_TARGET := ROCK_TARG_DIST / ROCK_VELOC_VECT;
--
--  Extrapolate target position based on TIME_TO_TARGET.
--  Since TIME_TO_TARGET does not take into account rocket accelaration,
--  it tends to be way off during low rocket velocities.  To reduce the
--  effect of this, limit the extrapolation to a reasonable period.
--
    if TIME_TO_TARGET > furthest_extrapolate then
      TIME_TO_TARGET := furthest_extrapolate;
    end if;
    EXTRAP_TARG.X := TARGET_POS.X + TARGET_VELOC_1.X * TIME_TO_TARGET;
    EXTRAP_TARG.Y := TARGET_POS.Y + TARGET_VELOC_1.Y * TIME_TO_TARGET;
--
--  prevent from extrapolating the target behind the rocket
--
    if EXTRAP_TARG.Y < ROCKET_POS.Y then
      EXTRAP_TARG.Y := ROCKET_POS.Y;
    end if;
    EXTRAP_TARG.Z := TARGET_POS.Z + TARGET_VELOC_1.Z * TIME_TO_TARGET;
```

```
  else
    TIME_TO_TARGET := integration_interval + 1.0;
    EXTRAP_TARG.X :=  TARGET_POS.X;
    EXTRAP_TARG.Y :=  TARGET_POS.Y;
    EXTRAP_TARG.Z :=  TARGET_POS.Z;
  end if;

  if TIME_TO_TARGET < integration_interval then
    INTEGRATION_PERIOD := TIME_TO_TARGET / Types.WORD(2);
    if INTEGRATION_PERIOD < 1.0 then
      INTEGRATION_PERIOD := 1.0;
    end if;
    INTEGRATION_SQ := (INTEGRATION_PERIOD-1.0) * (INTEGRATION_PERIOD-1.0);
  else
    INTEGRATION_PERIOD := integration_interval;
    INTEGRATION_SQ     := integration_int_sq;
  end if;
--
--  Compute where the ROCKET will be at the end of the INTEGRATION period.
--  All velocities will be calculated for that point to target.  Limit the
--  extrapolations to reasonable values.
--
  EXTRAP_ROCK.X := ROCKET_POS.X + ROCKET_VELOC_1.X * (INTEGRATION_PERIOD-1.0) +
                  (ROCKET_ACCEL.X / Types.WORD(2)) * INTEGRATION_SQ;
  if EXTRAP_ROCK.X > limit_rock_extrap then
    EXTRAP_ROCK.X := limit_rock_extrap;
  end if;
  EXTRAP_ROCK.Y := ROCKET_POS.Y + ROCKET_VELOC_1.Y * (INTEGRATION_PERIOD-1.0) +
                  (ROCKET_ACCEL.X / Types.WORD(2)) * INTEGRATION_SQ;
  if EXTRAP_ROCK.Y   limit_rock_extrap then
    EXTRAP_ROCK.Y := limit_rock_extrap;
  end if;
  EXTRAP_ROCK.Z := ROCKET_POS.Z + ROCKET_VELOC_1.Z * (INTEGRATION_PERIOD-1.0) +
                  (ROCKET_ACCEL.X / Types.WORD(2)) * INTEGRATION_SQ;
  if EXTRAP_ROCK.Z > limit_rock_extrap then
    EXTRAP_ROCK.Z := limit_rock_extrap;
  end if;

  ROCK_TARG_DELTA.X :=  EXTRAP_TARG.X - EXTRAP_ROCK.X;
  ROCK_TARG_DELTA.Y :=  EXTRAP_TARG.Y - EXTRAP_ROCK.Y;
  ROCK_TARG_DELTA.Z :=  EXTRAP_TARG.Z - EXTRAP_ROCK.Z;

  ROCK_TARG_DSQ_X := ROCK_TARG_DELTA.X * ROCK_TARG_DELTA.X;
  ROCK_TARG_DSQ_Y := ROCK_TARG_DELTA.Y * ROCK_TARG_DELTA.Y;
  ROCK_TARG_DSQ_Z := ROCK_TARG_DELTA.Z * ROCK_TARG_DELTA.Z;

  SUM := ROCK_TARG_DSQ_X + ROCK_TARG_DSQ_Y + ROCK_TARG_DSQ_Z;
  ROCK_TARG_DIST := Math.Sqrt(SUM);
  ROCK_TARG_XY_DIST := Math.Sqrt(ROCK_TARG_DSQ_X + ROCK_TARG_DSQ_Y);

--
```

```
--  Compute Desired Velocities in each axis for the end of INTEGRATION period.
--  If distance to target is too small to measure, then don't bother to find a
--  new desired velocity or acceleration because the rocket has already hit
--  the target by now!
--

  if ROCK_TARG_XY_DIST /= 0.0 then
    DESIRED_VELOC.X := ROCK_XY_VELOC_VECT *
                       (ROCK_TARG_DELTA.X / ROCK_TARG_XY_DIST);
    DESIRED_VELOC.Y := ROCK_XY_VELOC_VECT *
                       (ROCK_TARG_DELTA.Y / ROCK_TARG_XY_DIST);
    DESIRED_VELOC.Z := ROCK_VELOC_VECT * (ROCK_TARG_DELTA.Z / ROCK_TARG_DIST);

--
--  Compute Desired Accelerations
--

    DESIRED_ACCEL.X := (DESIRED_VELOC.X - ROCKET_VELOC_1.X) / INTEGRATION_PERIOD;
    DESIRED_ACCEL.Y := (DESIRED_VELOC.Y - ROCKET_VELOC_1.Y) / INTEGRATION_PERIOD;
    DESIRED_ACCEL.Z := (DESIRED_VELOC.Z - ROCKET_VELOC_1.Z) / INTEGRATION_PERIOD;

--
--  Compare Current Rocket Acceleration to Desired Rocket Acceleration
--  to produce Change in Acceleration
--
  end if;
  CHANGE_ACCEL.X := DESIRED_ACCEL.X - ROCKET_ACCEL.X *
              Math.SIN(Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT.AZIMUTH);
--
--  LIMIT THE CHANGE IN ACCELERATION
--
  if abs CHANGE_ACCEL.X > max_change then
    if CHANGE_ACCEL.X < 0.0 then
      CHANGE_ACCEL.X := -max_change;
    else
      CHANGE_ACCEL.X := max_change;
    end if;
  end if;

  CHANGE_ACCEL.Z := DESIRED_ACCEL.Z - ROCKET_ACCEL.Z;
  if abs CHANGE_ACCEL.Z > max_change then
    if CHANGE_ACCEL.Z < 0.0 then
      CHANGE_ACCEL.Z := -max_change;
    else
      CHANGE_ACCEL.Z := max_change;
    end if;
  end if;
--
--  Now translate from acceleration change requests to new aimpoint
--
  Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT :=
```

```
      AIMPOINT(Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT,CHANGE_ACCEL);
--
--  Now check if BOOST PHASE, if so go UP by adjusting ELEVATION.  Do not adjust
-- AZIMUTH because it is already pointing in the correct direction.
--
  if Aim_Data.ROCKET_INFO(ROCKET_ID).BOOST_PHASE then
    BOOST_LIMIT := ROCK_TARG_XY_DIST / Types.WORD(height_factor);
    if ROCKET_POS.Z > BOOST_LIMIT then
      Aim_Data.ROCKET_INFO(ROCKET_ID).BOOST_PHASE := FALSE;
    end if;
    Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT.ELEVATION :=
                                                   Config.launch_elevation;
  end if;                                    -- boost_phase check
  NEW_AIMPOINT := Aim_Data.ROCKET_INFO(ROCKET_ID).OLD_AIMPOINT;
end Guide;
```

```
------------------------------------------------------------------
--% UNIT:     Guide_Buf Task Subunit                             --
--% Effects:  Provides asynchronous comm. between simulator and Control.--
--% Modifies: No global data is modified.                        --
--% Requires: No initialization is required.                     --
--% Raises:   No explicitly raised exceptions are propagated.    --
--% Engineer: T. Griest.                                         --
------------------------------------------------------------------

--|
--| TASK BODY : Simulate.RDL.Guide_Buf
--|
--|   The Guide_Buf task acts as a buffer between the rocket data link
--| support task Rock_Sup and the Rocket.Control task which processes
--| the rocket data.  The direction flow is from Rocket.Control to the
--| Rock_Sup task, even though there are only accept statements here.  This is
--| to ease timing constraints.  The purpose of MSG_COUNT is to allow Rock_Sup
--| to use previous guidance messages if Rocket.Control is late sending it's
--| new guidance message.  However, it is set to zero at the start so
--| that before the main procedure gets a chance to run, Rock_Sup will
--| wait at the accept for at least one current guidance message, the
--| first one.  After the first guidance message is received, because
--| timing of the system is derived from Rock_Sup, Rock_Sup no longer will
--| need to wait for a new guidance message from Control.  This operation
--| reflects the fact that the rockets will continue to travel through space
--| regardless of whether there is guidance for them or not.
--|

-- Modifications Log
--
-- 88-10-20 : TEG => Original created.
--

with Debug_IO;
with Time_Stamp;

separate (Simulate.RDL)

task body Guide_Buf_Type is

  use Types;
  start       : constant Types.WORD_INDEX := 1; -- start of arrays
  GUIDE_MSG   : Rocket.ROCKET_GUIDE_MSG_TYPE;
  MSG_COUNT   : Types.WORD := 0;  -- if a message has been buffered
begin
  loop
    Time_Stamp.Log(0040);      --$TP(0040) Guidebuf task start
    select
      accept Put_Guide(DATA : in Rocket.ROCKET_GUIDE_MSG_TYPE) do
        Time_Stamp.Log(0041);      --$TP(0041) Guidebuf accept Put_Guide start
        GUIDE_MSG.NUM_ROCKETS := DATA.NUM_ROCKETS;          -- copy data
```

```
      GUIDE_MSG.ROCKET_GUIDE_LIST(start..DATA.NUM_ROCKETS) :=
                      DATA.ROCKET_GUIDE_LIST(start..DATA.NUM_ROCKETS);
      MSG_COUNT := 1;          -- only meaningful that it is > 0
      Time_Stamp.Log(0042);     --$TP(0042) Guidebuf accept Put_Guide end
    end Put_Guide;
  or
    when MSG_COUNT > 0 =>
    accept Get_Guide(DATA : out Rocket.ROCKET_GUIDE_MSG_TYPE) do
      Time_Stamp.Log(0043);     --$TP(0043) Guidebuf accept Get_Guide start
      DATA.NUM_ROCKETS := GUIDE_MSG.NUM_ROCKETS;
      DATA.ROCKET_GUIDE_LIST(start..GUIDE_MSG.NUM_ROCKETS) :=
           GUIDE_MSG.ROCKET_GUIDE_LIST(start..GUIDE_MSG.NUM_ROCKETS);
      MSG_COUNT := 1;             -- do keep multiple copies
      Time_Stamp.Log(0044);     --$TP(0044) Guidebuf accept Get_Guide end
    end Get_Guide;
  end select;
 Time_Stamp.Log(0045);     --$TP(0045) Guidebuf task end
 end loop;
exception
 when others =>
   Debug_IO.Put_Line("GUIDE_BUF termination due to exception.");
end Guide_Buf_Type;
```

```
--------------------------------------------------------------------------
--% UNIT        : Hardware Configuration Spec.                            --
--% Effects     : None.                                                   --
--% Modifies    : Nothing.                                                --
--% Requires    : The hardware defined below.                             --
--% Raises      : No exceptions.                                          --
--% Engineer    : M. Sperry.                                              --
--------------------------------------------------------------------------

--|
--| PACKAGE SPEC : HW_Config
--|
--|   This package is designed to familiarize the user with the hardware that
--| the BDS was originally implemented upon.  It is implemented on a TANDY 4000
--| with an EGA screen, utilizing a Logitech C7 Serial Mouse on serial port COM2
--| as a pointing device.  The timer chip addresses and values are defined.
--| Note : Some machine addresses (for the EGA especially) are in the package
--| Machine_Dependent.
--|

-- Modifications Log
--
-- 89-08-08 : MPS => Original created.
-- 89-11-19 : MPS => Added timer chip addresses and constants
--

with Types;
with Low_Level_IO;

package HW_Config is

-- The following addresses are used for this machine.

COM2_data          : constant Low_Level_IO.PORT_ADDRESS := 16#2F8#;
COM2_int_enable    : constant Low_Level_IO.PORT_ADDRESS := 16#2F9#;
COM2_int_ident     : constant Low_Level_IO.PORT_ADDRESS := 16#2FA#;
COM2_control       : constant Low_Level_IO.PORT_ADDRESS := 16#2FB#;
COM2_modem_control : constant Low_Level_IO.PORT_ADDRESS := 16#2FC#;
COM2_status        : constant Low_Level_IO.PORT_ADDRESS := 16#2FD#;

pic_8259           : constant Low_Level_IO.PORT_ADDRESS := 16#20#;
pic_8259_mr        : constant Low_Level_IO.PORT_ADDRESS := 16#21#;

counter_two_addr   : constant := 16#42#;
timer_control_addr : constant := 16#43#;

end HW_Config;
```

```
--------------------------------------------------------------------------
--% UNIT:     Interrupt_Control Package Spec. and Body.                   --
--% Effects:  Provides control over interrupt flags.                      --
--% Modifies: No global data is modified.                                 --
--% Requires: No initialization is required.                              --
--% Raises:   No explicitly raised exceptions are propagated.             --
--% Engineer: M. Sperry.                                                  --
--------------------------------------------------------------------------

--|
--| PACKAGE SPEC : Interrupt_Control
--|
--|   The purpose of the Interrupt_Control package is to provide Ada level
--| semantics for disabling and enabling interrupts on the 80X86 family of
--| processors.  Also for clearing the direction flag because of an RTE bug
--| which does not always clear it.
--|

-- Modifications Log
--
-- 88-11-20 : MPS => Original created.
--

with Machine_Code;
use Machine_Code;
pragma ELABORATE(Machine_Code);

package Interrupt_Control is

pragma SUPPRESS(Elaboration_Check);

procedure Disable;
pragma INLINE(Disable);
procedure Enable;
pragma INLINE(Enable);

procedure Clear_Direction_Flag;
pragma INLINE(Clear_Direction_Flag);

end Interrupt_Control;

--|
--| PACKAGE BODY : Interrupt_Control
--|
--|   Interrupt_Control is implemented with machine code statments.  The
--| suppression of the elaboration check is used to make the inlining of
--| these machine instructions as short as possible.
--|

package body Interrupt_Control is
```

```
procedure Disable is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
end Disable;

procedure Enable is
begin
  MACHINE_INSTRUCTION'(none,m_STI);
end Enable;

procedure Clear_Direction_Flag is
begin
  MACHINE_INSTRUCTION'(none,m_CLD);
end Clear_Direction_Flag;

end Interrupt_Control;
```

```
----------------------------------------------------------------------
--% UNIT:     Machine_Dependent Package Spec.                           --
--% Effects:  Provides machine dependent operations for enhanced speed. --
--% Modifies: No global data is modified.                               --
--% Requires: Graphics mode, and initialization of timer channel two.   --
--% Raises:   No explicitly raised exceptions are propagated.           --
--% Engineer: M. Sperry.                                                --
----------------------------------------------------------------------

--|
--| PACKAGE SPEC : Machine_Dependent
--|
--|   Package Machine_Dependent contains machine code statements to perform
--| low level graphics functions, including an interface to the BIOS routines
--| found on the EGA (for text processing).  Note that these instructions
--| are inlined to enhance speed.
--|   Also implemented are routines which perform fixed point multiplications
--| and divisions in machine code for speed enhancements.
--|   And, a procedure which returns the value in the channel two counter.
--|

--
-- Modifications Log
--
-- 88-11-04 : MPS => Original created.
-- 89-08-24 : MPS => Specifications for fixed math routines incorporated.
-- 89-11-21 : MPS => Next_Random created.
--

with Machine_Code;
with Graphics;
with Types;
use Machine_Code;
pragma ELABORATE(Machine_Code);

package Machine_Dependent is

start_countdown  : constant := 16#B2#;          -- mode 2, channel 2
max_timer_value  : constant := 256;             -- channel 2 LSB divisor

procedure Put_Pixel(ABS_X, ABS_Y : Types.COORDINATE;
                    COLOR        : Graphics.COLOR_TYPE);
pragma INLINE(Put_Pixel);

procedure Write_Mode_0;
pragma INLINE(Write_Mode_0);

--
--  Provide a mechanism to call ROM located routine to initialize screen
--
procedure Int10(BIOS_FUNCTION : Types.WORD;-- spec to BIOS graphics call
```

```
              PARAM_1        : Types.WORD;
              PARAM_2        : Types.WORD);
pragma INTERFACE(ASM86, Int10);
pragma INTERFACE_SPELLING(Int10, "D1BIOS?GRAPHICSCALL");

procedure Write_Mode_2;
pragma INLINE(Write_Mode_2);

procedure Long_Long_Mul(LEFT,RIGHT : Types.LONG_FIXED;
                        RESULT     : out Types.LONG_FIXED);
pragma INLINE(Long_Long_Mul);

procedure Long_Long_Div(LEFT,RIGHT : Types.LONG_FIXED;
                        RESULT     : out Types.LONG_FIXED);
pragma INLINE(Long_Long_Div);

procedure Long_Word_Div(LEFT   : Types.LONG_FIXED;
                        RIGHT  : Types.WORD;
                        RESULT : out Types.LONG_FIXED);
pragma INLINE(Long_Word_Div);

procedure Meters_Meters_Div(LEFT,RIGHT : Types.METERS;
                            RESULT     : out Types.METERS);
pragma INLINE(Meters_Meters_Div);

procedure Meters_Word_Div(LEFT   : Types.METERS;
                          RIGHT  : Types.WORD;
                          RESULT : out Types.METERS);
pragma INLINE(Meters_Word_Div);

procedure Meters_Meters_Mul(LEFT,RIGHT : Types.METERS;
                            RESULT     : out Types.METERS);
pragma INLINE(Meters_Meters_Mul);

procedure Rate_Rate_Mul(LEFT,RIGHT : Types.RATE_TYPE;
                        RESULT     : out Types.RATE_TYPE);
pragma INLINE(Rate_Rate_Mul);

procedure Rate_Rate_Div(LEFT,RIGHT : Types.RATE_TYPE;
                        RESULT     : out Types.RATE_TYPE);
pragma INLINE(Rate_Rate_Div);

procedure Rate_Word_Div(LEFT   : Types.RATE_TYPE;
                        RIGHT  : Types.WORD;
                        RESULT : out Types.RATE_TYPE);
pragma INLINE(Rate_Word_Div);

procedure Long_Rate_Div(LEFT   : Types.LONG_FIXED;
                        RIGHT  : Types.RATE_TYPE;
                        RESULT : out Types.LONG_FIXED);
pragma INLINE(Long_Rate_Div);
```

```
procedure Rate_Long_Div(LEFT   : Types.RATE_TYPE;
                        RIGHT  : Types.LONG_FIXED;
                        RESULT : out Types.RATE_TYPE);
pragma INLINE(Rate_Long_Div);

procedure Rate_Long_Long_Mul(LEFT   : Types.RATE_TYPE;
                             RIGHT  : Types.LONG_FIXED;
                             RESULT : out Types.LONG_FIXED);
pragma INLINE(Rate_Long_Long_Mul);

procedure Rate_Long_Rate_Mul(LEFT   : Types.RATE_TYPE;
                             RIGHT  : Types.LONG_FIXED;
                             RESULT : out Types.RATE_TYPE);
pragma INLINE(Rate_Long_Rate_Mul);

procedure Next_Random(CHANNEL_TWO_VALUE : out Types.WORD_INDEX);
pragma INLINE(Next_Random);

end Machine_Dependent;
```

```
-----------------------------------------------------------------
--% UNIT:     Machine_Dependent Package Body.                   --
--% Effects:  Provides graphics machine dependencies.           --
--% Modifies: No global data is modified.                       --
--% Requires: No initialization is required (other than graphics mode). --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: M. Sperry.                                        --
-----------------------------------------------------------------

--|
--| PACKAGE BODY : Machine_Dependent
--|
--|   A package which makes use of the functionality of the BIOS routines
--| found in an EGA card to perform some graphics processing.  Note that
--| some register level EGA programming is performed.
--|   Also, the timer chip channel two functions are utilized to generate
--| pseudo random numbers.
--|

-- Modifications Log
--
-- 88-08-25 : MPS => Original created.
-- 89-08-24 : MPS => Incorporated fixed math routine bodies for speed.
-- 89-11-28 : MPS => developed Next_Random procedure.
--

with HW_Config;

package body Machine_Dependent is

hi_res_graphics  : constant := 16#10#;          -- graphics mode
set_cursor       : constant := 16#0200#;        -- set cursor function
page_zero        : constant := 16#00#;          -- set cursor to active page
write_function   : constant := 16#0E#;
index_register   : constant := 16#3CE#;         -- port address
access_register  : constant := 16#3CF#;         -- port address
mode_register    : constant := 5;               -- index register 5
write_mode_2_val : constant := 2;
write_mode_0_val : constant := 0;



procedure Put_Pixel(ABS_X, ABS_Y : Types.COORDINATE;
                    COLOR        : Graphics.COLOR_TYPE) is

--|
--| SUBPROGRAM BODY : Machine_Dependent.Put_Pixel
--|
--|   An assembly level procedure (for enhanced speed) to place a dot on the EGA
--| screen.  Write mode two is used here (again, for enhanced speed).  It is
--| important to note that this routine could be called up to 1235 times per
```

```
--| interval.
--|

begin
--
-- The first thing to do is find out which bit must be turned on.  This is
-- done by taking SHR( 80h, ABS_X mod 8 ).  The bit ordering goes from 7 -> 0.
--
  MACHINE_INSTRUCTION'(register_register, m_MOV, CX, CX); -- defeat compiler bug

  MACHINE_INSTRUCTION'(register_immediate, m_MOV, DX, 16#3CE#);  -- select bit
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, 8);  -- mask register
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);  -- in graphics chip
--
-- Determine which bit must be turned on.  This is
-- done by taking SHR( 80h, ABS_X rem 8 ), reversing the bit ordering.
--
  MACHINE_INSTRUCTION'(register_system_address. m_MOV, CX, ABS_X'address); --X
  MACHINE_INSTRUCTION'(register_register, m_MOV, BX, CX);   -- make copy of X
  MACHINE_INSTRUCTION'(register_immediate, m_AND, CL, 7);   -- mask for bit #
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, 16#80#);  -- most significant bit is
  MACHINE_INSTRUCTION'(register_register, m_SHR, AL, CL); -- bit zero, do bit reversal.
--
-- AL now holds the bit mask.  Now give it to the bit mask register located
-- at 16#3CF#.
--
  MACHINE_INSTRUCTION'(register, m_INC, DX);  -- increment port address to 3CF
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);

--
-- Now, latch the byte of graphics memory.  The byte to latch
-- is defined as (ABS_Y * 80) + (ABS_X / 8).  Then, when giving
-- it back, place the color in AL.  Note that only four bits of the color are
-- significant and that the color placed in AL is not actually a color, but a
-- palette register selection (from 0 to 15).  The color in the palette
-- register is the color displayed.-16#6000# is loaded (= A000H)
-- to point to the EGA graphics page zero memory address.
--
  MACHINE_INSTRUCTION'(register_system_address, m_MOV, AX, ABS_Y'address);--Y
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, CX, 80); -- bytes/line
  MACHINE_INSTRUCTION'(register, m_MUL, CX);      -- ABS_Y * 80 in AX
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, CL, 3);  -- Shift Count
  MACHINE_INSTRUCTION'(register_register, m_SHR, BX, CL);  -- ABS_X / 8 in BX
  MACHINE_INSTRUCTION'(register_register, m_ADD, BX, AX);  -- BX is offset
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AX, -16#6000#); -- base of RAM
  MACHINE_INSTRUCTION'(register_register, m_MOV, ES, AX);
--
-- Latch the palette selection.  Note that the contents of AL upon return are
-- meaningless, and that the color is latched internally to the EGA's four bit
-- planes.
--
```

```
-- mov AL,ES:[BX]
  MACHINE_INSTRUCTION'(register_address, m_MOV, AL, ES, BX, nil, SCALE_1, 0);
  MACHINE_INSTRUCTION'(register_system_address, m_MOV, AX, COLOR'address);
--
-- Finally, give the palette selection (color) to the four bit planes.
--
-- mov ES:[BX],AL
  MACHINE_INSTRUCTION'(address_register, m_MOV, ES, BX, nil, SCALE_1, 0, AL);

end Put_Pixel;

procedure Write_Mode_0 is

--|
--| SUBPROGRAM BODY : Machine_Dependent.Write_Mode_0
--|
--|   A procedure used to change the write mode of the screen to mode 0,
--| for text writing.  This procedure is called before writing any text.
--|

begin
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, DX, index_register);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, mode_register);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, DX, access_register);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, write_mode_0_val);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
end Write_Mode_0;

procedure Write_Mode_2 is

--|
--| SUBPROGRAM BODY : Machine_Dependent.Write_Mode_2
--|
--|   A procedure used to change the write mode of the screen to mode 2, which
--| facilitates the process of pixel plotting.  This routine is called after
--| writing the necessary statistics titles, etc.
--|

begin
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, DX, index_register);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, mode_register);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, DX, access_register);
  MACHINE_INSTRUCTION'(register_immediate, m_MOV, AL, write_mode_2_val);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
end Write_Mode_2;

--|
--| SUBPROGRAM BODY : Machine_Dependent.Fixed_Math routines
--|
```

```
--| These routines are written in 80386 32-bit code optimized for the
--| types on which they operate for maximum speed.  They are treated as a
--| resource.
--|

procedure Long_Long_Mul(LEFT,RIGHT : Types.LONG_FIXED;
                        RESULT     : out Types.LONG_FIXED) is

begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address,m_IMUL,RIGHT'address);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,6);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHRD EAX,EDX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#AD#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#D0#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Long_Long_Mul;

procedure Long_Long_Div(LEFT,RIGHT : Types.LONG_FIXED;
                        RESULT     : out Types.LONG_FIXED) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(none,m_CWD);  -- CDQ
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,6);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHLD EDX,EAX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#A5#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#C2#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_SHL,AX,CL);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address,m_IDIV,RIGHT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Long_Long_Div;
```

```
procedure Long_Word_Div(LEFT   : Types.LONG_FIXED;
                        RIGHT  : Types.WORD;
                        RESULT : out Types.LONG_FIXED) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,RIGHT'address);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(none,m_CBW);     -- this instruction performs a CWDE
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_MOV,BX,AX);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,SI,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,SI,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(none,m_CWD);     -- this instruction performs a CDQ
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register,m_IDIV,BX);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Long_Word_Div;

procedure Meters_Meters_Mul(LEFT,RIGHT : Types.METERS;
                            RESULT     : out Types.METERS) is
begin
  MACHINE_INSTRUCTION'(register_register, m_MOV, CX, CX); -- defeat compiler bug
  MACHINE_INSTRUCTION'(register_system_address,m_MOV,AX,LEFT'address);
  MACHINE_INSTRUCTION'(system_address,m_IMUL,RIGHT'address);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,3);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHRD AX,DX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#AD#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#D0#);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
end Meters_Meters_Mul;

procedure Meters_Meters_Div(LEFT,RIGHT : Types.METERS;
                            RESULT     : out Types.METERS) is
begin
  MACHINE_INSTRUCTION'(register_register, m_MOV, CX, CX); -- defeat compiler bug
  MACHINE_INSTRUCTION'(register_system_address,m_MOV,AX,LEFT'address);
  MACHINE_INSTRUCTION'(none,m_CWD);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,3);
  MACHINE_INSTRUCTION'(register_register,m_SHL,AX,CL);
  MACHINE_INSTRUCTION'(system_address,m_IDIV,RIGHT'address);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
end Meters_Meters_Div;

procedure Meters_Word_Div(LEFT   : Types.METERS;
                          RIGHT  : Types.WORD;
                          RESULT : out Types.METERS) is
```

```
begin
  MACHINE_INSTRUCTION'(register_register, m_MOV, CX, CX); -- defeat compiler bug
  MACHINE_INSTRUCTION'(register_system_address,m_MOV,AX,LEFT'address);
  MACHINE_INSTRUCTION'(none,m_CWD);
  MACHINE_INSTRUCTION'(register_system_address,m_MOV,BX,RIGHT'address);
  MACHINE_INSTRUCTION'(register,m_IDIV,BX);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
end Meters_Word_Div;

procedure Rate_Rate_Mul(LEFT,RIGHT : Types.RATE_TYPE;
                        RESULT     : out Types.RATE_TYPE) is

begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address,m_IMUL,RIGHT'address);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,16);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHRD EAX,EDX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#AD#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#D0#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Rate_Mul;

procedure Rate_Rate_Div(LEFT,RIGHT : Types.RATE_TYPE;
                        RESULT     : out Types.RATE_TYPE) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(none,m_CWD);               -- performs a CDQ
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,16);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHLD EDX,EAX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#A5#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#C2#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_SHL,AX,CL);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(system_address,m_IDIV,RIGHT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Rate_Div;
```

```
procedure Rate_Word_Div(LEFT   : Types.RATE_TYPE;
                        RIGHT  : Types.WORD;
                        RESULT : out Types.RATE_TYPE) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,RIGHT'address);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(none,m_CBW);    -- this instruction performs a CWDE
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_MOV,BX,AX);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,SI,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,SI,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(none,m_CWD);    -- this instruction performs a CDQ
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register,m_IDIV,BX);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Word_Div;

procedure Rate_Long_Long_Mul(LEFT   : Types.RATE_TYPE;
                             RIGHT  : Types.LONG_FIXED;
                             RESULT : out Types.LONG_FIXED) is

begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address,m_IMUL,RIGHT'address);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,16);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHRD EAX,EDX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#AD#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#D0#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Long_Long_Mul;

procedure Rate_Long_Rate_Mul(LEFT   : Types.RATE_TYPE;
                             RIGHT  : Types.LONG_FIXED;
                             RESULT : out Types.RATE_TYPE) is

begin
  MACHINE_INSTRUCTION'(none,m_CLI);
```

```
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address,m_IMUL,RIGHT'address);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,6);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHRD EAX,EDX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#AD#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#D0#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Long_Rate_Mul;

procedure Long_Rate_Div(LEFT   : Types.LONG_FIXED;
                        RIGHT  : Types.RATE_TYPE;
                        RESULT : out Types.LONG_FIXED) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(none,m_CWD);               -- performs a CDQ
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,16);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHLD EDX,EAX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#A5#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#C2#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_SHL,AX,CL);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(system_address,m_IDIV,RIGHT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Long_Rate_Div;

procedure Rate_Long_Div(LEFT   : Types.RATE_TYPE;
                        RIGHT  : Types.LONG_FIXED;
                        RESULT : out Types.RATE_TYPE) is
begin
  MACHINE_INSTRUCTION'(none,m_CLI);
  MACHINE_INSTRUCTION'(register_system_address,m_LEA,BX,LEFT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_address,m_MOV,AX,SS,BX,nil,scale_1,0);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(none,m_CWD);               -- performs a CDQ
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,CX,6);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
```

```
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#0F#); -- SHLD EDX,EAX,CL
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#A5#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#C2#);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(register_register,m_SHL,AX,CL);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#);
  MACHINE_INSTRUCTION'(system_address,m_IDIV,RIGHT'address);
  MACHINE_INSTRUCTION'(immediate,m_DATAB,16#66#); -- 32 bit override
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,RESULT'address,AX);
  MACHINE_INSTRUCTION'(none,m_STI);
end Rate_Long_Div;

procedure Next_Random(CHANNEL_TWO_VALUE : out Types.WORD_INDEX) is

--|
--| SUBPROGRAM BODY : Machine_Dependent.Next_Random
--|
--|   This function returns the value of the number at the address of the
--| channel two counter.  It is assumed that the initialization of that channel
--| is previously performed.  Therfore it returns a value between 0 and 255.
--|

begin
  MACHINE_INSTRUCTION'(register_register, m_MOV, CX, CX); -- defeat compiler bug
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,DX,HW_Config.counter_two_addr);
  MACHINE_INSTRUCTION'(register_register,m_IN,AL,DX);
  MACHINE_INSTRUCTION'(register_register,m_XOR,AH,AH);
  MACHINE_INSTRUCTION'(system_address_register,m_MOV,CHANNEL_TWO_VALUE'address,
                       AX);
end Next_Random;

procedure Initialize_Timer_Two is

--|
--| SUBPROGRAM BODY : Machine_Dependent.Initialize_Timer_Two
--|
--|   A procedure used to start the channel 2 counter counting down from
--| max_timer_value to zero over and over to generate numbers.
--|

begin
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,DX,
                       HW_Config.timer_control_addr);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,AX,start_countdown);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,DX,HW_Config.counter_two_addr);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,AX,max_timer_value);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
--
-- The first out byte was the LSB. Now take care of MSB.
--
```

```
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,DX,HW_Config.counter_two_addr);
  MACHINE_INSTRUCTION'(register_immediate,m_MOV,AX,max_timer_value);
  MACHINE_INSTRUCTION'(register_register, m_OUT, DX, AL);
end Initialize_Timer_Two;
pragma INLINE(Initialize_Timer_Two);

begin
  Initialize_Timer_Two;
end Machine_Dependent;
```

```
-----------------------------------------------------------------
--% UNIT:     Math Package Spec.                                --
--% Effects:  Compute various functions: Tan, Arc Tan, and Sqrt.  --
--% Modifies: No global data is modified.                        --
--% Requires: No initialization is required.                     --
--% Raises:   No explicitly raised exceptions are propagated.    --
--% Engineer: Various.                                           --
-----------------------------------------------------------------

--|
--| PACKAGE SPEC : Math
--|
--|   Math is responsible for implementing the necessary math functions of
--| the BDS.  Some of the transcedental functions are approximations using
--| table look-ups.  The fixed point routines which perform basic multiply
--| and divide operations are made visible to replace the runtime supplied
--| routines which are too general (and thus too slow) for the BDS's needs.
--| The Get_Random_Num function uses the channel two timer to return a
--| number from 0 to the limit specified.
--|

--
-- Modifications Log
--
-- 88-10-09 : TEG => Original created.
-- 89-08-24 : MPS => Fixed point routines were made visible.
-- 89-09-11 : MPS => Sin,Cos added from intern code.
-- 89-11-08 : MPS => General Power routine added.
-- 89-11-08 : MPS => Arcsin routine added.
-- 89-11-08 : LJG => Arctan routine was made more precise.
-- 89-11-20 : MPS => Get_Random_Num routine moved from Targ_Sup to Math.
--

with Types;

package Math is

function "*"(LEFT,RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED;
--pragma INLINE("*");

function "/"(LEFT,RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED;
--pragma INLINE("/");

function "/"(LEFT  : Types.LONG_FIXED;
             RIGHT : Types.WORD) return Types.LONG_FIXED;
--pragma INLINE("/");

function "*"(LEFT,RIGHT : Types.METERS) return Types.METERS;
--pragma INLINE("*");

function "/"(LEFT,RIGHT : Types.METERS) return Types.METERS;
```

```
--pragma INLINE("/");

function "/"(LEFT  : Types.METERS;
             RIGHT : Types.WORD) return Types.METERS;
--pragma INLINE("/");

function "*"(LEFT,RIGHT : Types.RATE_TYPE) return Types.RATE_TYPE;
--pragma INLINE("*");

function "/"(LEFT,RIGHT : Types.RATE_TYPE) return Types.RATE_TYPE;
--pragma INLINE("/");

function "/"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.WORD) return Types.RATE_TYPE;
--pragma INLINE("/");

function "*"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.RATE_TYPE;
--pragma INLINE("*");

function "*"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED;
--pragma INLINE("*");

function "/"(LEFT  : Types.LONG_FIXED;
             RIGHT : Types.RATE_TYPE) return Types.LONG_FIXED;
--pragma INLINE("/");

function "/"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.RATE_TYPE;
--pragma INLINE("/");

function Power (BASE, RAISED_TO : Types.LONG_FIXED) return Types.LONG_FIXED;

function Sin (ANGLE : Types.BAM) return Types.LONG_FIXED;

function Cos (ANGLE : Types.BAM) return Types.LONG_FIXED;

function Tan (ANGLE : Types.BAM) return Types.LONG_FIXED;

function Arcsin (THETA : Types.LONG_FIXED) return Types.LONG_FIXED;-- in degrees

function Arctan (Z_INPUT : Types.LONG_FIXED) return Types.BAM;

function Sqrt (X : in Types.METERS) return Types.METERS;

function Sqrt (X : in Types.LONG_FIXED) return Types.LONG_FIXED;

function Sqrt (X : in Types.RATE_TYPE) return Types.RATE_TYPE;

function Get_Random_Num (LIMIT : Types.WORD_INDEX) return Types.WORD_INDEX;
```

```
function Get_Random_Num (LIMIT : Types.METERS) return Types.METERS;

function Get_Random_Num (LIMIT : Types.LONG_FIXED) return Types.LONG_FIXED;

end Math;
```

```
------------------------------------------------------------------------
--% UNIT:     Math Package Body.                                        --
--% Effects:  Compute various functions: Tan, Arc Tan, and Sqrt.       --
--% Modifies: No global data is modified.                               --
--% Requires: No initialization is required.                            --
--% Raises:   No explicitly raised exceptions are propagated.           --
--% Engineer: L. Griest.                                                --
------------------------------------------------------------------------

--|
--| PACKAGE BODY : Math
--|
--|   The math package contains the various math routines needed by the BDS.
--| Some of these routines contain simplifications to increase performance.
--| The routines (provided to do fixed point math) are all functions which allow
--| for overloading.  Since machine code statements can only be procedures, a
--| call to the appropriate procedure is contained within each function.  Since
--| each function and procedure is inlined, the end result should not generate
--| any overhead.
--|

-- Modifications Log
--
-- 88-10-10 : TEG => Original created.
-- 89-08-24 : MPS => Bodies of fixed point functions created.
-- 89-11-08 : MPS => Arcsin and Power functions added.
-- 89-11-09 : LJG => Arctan function was given greater accuracy.
-- 89-11-20 : MPS => Get_Random_Num function created.
--

with Machine_Dependent;
with Time_Stamp;
  pragma ELABORATE(Time_Stamp, Machine_Dependent);

package body Math is
  use Types;

  type FUNC_NAME is ( SINE, COSINE);
  subtype FUNC_RANGE is Types.WORD range 0..100;
  type FUNC_TABLE is array (FUNC_RANGE,FUNC_NAME) of Types.LONG_FIXED;
  TRIG_FUNC : FUNC_TABLE  := ((0.000000,1.000000),(0.015625,1.000000),
                              (0.031250,1.000000),(0.046875,1.000000),
                              (0.062500,1.000000),(0.078125,1.000000),
                              (0.093750,1.000000),(0.109375,1.000000),
                              (0.125000,1.000000),(0.140625,1.000000),
                              (0.156250,1.000000),(0.171875,1.000000),
                              (0.187500,0.984375),(0.203125,0.984375),
                              (0.218750,0.984375),(0.234375,0.984375),
                              (0.250000,0.968750),(0.265625,0.968750),
                              (0.281250,0.968750),(0.296875,0.968750),
                              (0.312500,0.953125),(0.328125,0.953125),
```

```
                (0.343750,0.953125),(0.359375,0.937500),
                (0.375000,0.937500),(0.390625,0.937500),
                (0.406250,0.921875),(0.421875,0.921875),
                (0.437500,0.906250),(0.453125,0.906250),
                (0.453125,0.890625),(0.468750,0.890625),
                (0.484375,0.875000),(0.500000,0.875000),
                (0.515625,0.859375),(0.531250,0.859375),
                (0.546875,0.843750),(0.546875,0.843750),
                (0.562500,0.828125),(0.578125,0.828125),
                (0.593750,0.812500),(0.609375,0.796875),
                (0.625000,0.796875),(0.625000,0.781250),
                (0.640625,0.765625),(0.656250,0.765625),
                (0.671875,0.750000),(0.671875,0.734375),
                (0.687500,0.718750),(0.703125,0.718750),
                (0.703125,0.703125),(0.718750,0.703125),
                (0.734375,0.687500),(0.750000,0.687500),
                (0.750000,0.671875),(0.765625,0.656250),
                (0.765625,0.640625),(0.781250,0.625000),
                (0.796875,0.609375),(0.796875,0.593750),
                (0.812500,0.593750),(0.812500,0.578125),
                (0.828125,0.562500),(0.828125,0.546875),
                (0.843750,0.531250),(0.843750,0.515625),
                (0.859375,0.515625),(0.859375,0.500000),
                (0.875000,0.484375),(0.875000,0.468750),
                (0.890625,0.453125),(0.890625,0.437500),
                (0.890625,0.437500),(0.906250,0.421875),
                (0.921875,0.406250),(0.937500,0.390625),
                (0.937500,0.375000),(0.937500,0.359375),
                (0.937500,0.343750),(0.953125,0.343750),
                (0.953125,0.312500),(0.953125,0.296875),
                (0.953125,0.296875),(0.953125,0.281250),
                (0.968750,0.250000),(0.984375,0.250000),
                (0.984375,0.234375),(0.984375,0.218750),
                (0.984375,0.203125),(0.984375,0.187500),
                (0.984375,0.171875),(1.000000,0.156250),
                (1.000000,0.140625),(1.000000,0.125000),
                (1.000000,0.109375),(1.000000,0.093750),
                (1.000000,0.078125),(1.000000,0.062500),
                (1.000000,0.031250),(1.000000,0.015625),
                (1.000000,0.000000));

 TAN_TABLE  : array(Types.WORD range 0..90) of Types.LONG_FIXED :=
(
 0.00000,  0.01746,  0.03492,  0.05241,  0.06993,  0.08749,  0.10510,  0.12278,
 0.14054,  0.15838,  0.17633,  0.19438,  0.21256,  0.23087,  0.24933,  0.26795,
 0.28675,  0.30573,  0.32492,  0.34433,  0.36397,  0.38386,  0.40403,  0.42447,
 0.44523,  0.46631,  0.48773,  0.50953,  0.53171,  0.55431,  0.57735,  0.60086,
 0.62487,  0.64941,  0.67451,  0.70021,  0.72654,  0.75356,  0.78129,  0.80978,
 0.83910,  0.86929,  0.90040,  0.93252,  0.96569,  1.00000,  1.03553,  1.07237,
 1.11061,  1.15037,  1.19175,  1.23490,  1.27994,  1.32704,  1.37638,  1.42815,
 1.48256,  1.53986,  1.60033,  1.66428,  1.73205,  1.80405,  1.88073,  1.96261,
```

```
 2.05030,  2.14451,  2.24604,  2.35585,  2.47509,  2.60509,  2.74748,  2.90421,
 3.07768,  3.27085,  3.48741,  3.73205,  4.01078,  4.33148,  4.70463,  5.14455,
 5.67128,  6.31375,  7.11536,  8.14434,  9.51436, 11.43005, 14.30067, 19.08114,
28.63625, 57.28996, Types.sqrt_large_number);

function "*"(LEFT,RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED is
  RESULT         : Types.LONG_FIXED;
  MULTIPLICAND_1 : Types.LONG_FIXED := LEFT;
  MULTIPLICAND_2 : Types.LONG_FIXED := RIGHT;
begin
  Machine_Dependent.Long_Long_Mul(MULTIPLICAND_1,MULTIPLICAND_2,RESULT);
  return RESULT;
end "*";

function "/"(LEFT,RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED is
  QUOTIENT : Types.LONG_FIXED;
  DIVIDEND : Types.LONG_FIXED := LEFT;
  DIVISOR  : Types.LONG_FIXED := RIGHT;
begin
  Machine_Dependent.Long_Long_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "/"(LEFT  : Types.LONG_FIXED;
             RIGHT : Types.WORD) return Types.LONG_FIXED is
  QUOTIENT : Types.LONG_FIXED;
  DIVIDEND : Types.LONG_FIXED := LEFT;
  DIVISOR  : Types.WORD       := RIGHT;
begin
  Machine_Dependent.Long_Word_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "*"(LEFT,RIGHT : Types.METERS) return Types.METERS is
  RESULT         : Types.METERS;
  MULTIPLICAND_1 : Types.METERS := LEFT;
  MULTIPLICAND_2 : Types.METERS := RIGHT;
begin
  Machine_Dependent.Meters_Meters_Mul(MULTIPLICAND_1,MULTIPLICAND_2,RESULT);
  return RESULT;
end "*";

function "/"(LEFT,RIGHT : Types.METERS) return Types.METERS is
  QUOTIENT : Types.METERS;
  DIVIDEND : Types.METERS := LEFT;
  DIVISOR  : Types.METERS := RIGHT;
begin
  Machine_Dependent.Meters_Meters_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";
```

```
function "/"(LEFT  : Types.METERS;
             RIGHT : Types.WORD) return Types.METERS is
  QUOTIENT : Types.METERS;
  DIVIDEND : Types.METERS := LEFT;
  DIVISOR  : Types.WORD   := RIGHT;
begin
  Machine_Dependent.Meters_Word_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "*"(LEFT,RIGHT : Types.RATE_TYPE) return Types.RATE_TYPE is
  RESULT         : Types.RATE_TYPE;
  MULTIPLICAND_1 : Types.RATE_TYPE := LEFT;
  MULTIPLICAND_2 : Types.RATE_TYPE := RIGHT;
begin
  Machine_Dependent.Rate_Rate_Mul(MULTIPLICAND_1,MULTIPLICAND_2,RESULT);
  return RESULT;
end "*";

function "/"(LEFT,RIGHT : Types.RATE_TYPE) return Types.RATE_TYPE is
  QUOTIENT : Types.RATE_TYPE;
  DIVIDEND : Types.RATE_TYPE := LEFT;
  DIVISOR  : Types.RATE_TYPE := RIGHT;
begin
  Machine_Dependent.Rate_Rate_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "/"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.WORD) return Types.RATE_TYPE is
  QUOTIENT : Types.RATE_TYPE;
  DIVIDEND : Types.RATE_TYPE := LEFT;
  DIVISOR  : Types.WORD      := RIGHT;
begin
  Machine_Dependent.Rate_Word_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "/"(LEFT  : Types.LONG_FIXED;
             RIGHT : Types.RATE_TYPE) return Types.LONG_FIXED is
  QUOTIENT : Types.LONG_FIXED;
  DIVIDEND : Types.LONG_FIXED := LEFT;
  DIVISOR  : Types.RATE_TYPE := RIGHT;
begin
  Machine_Dependent.Long_Rate_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "/"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.RATE_TYPE is
  QUOTIENT : Types.RATE_TYPE;
```

```
  DIVIDEND : Types.RATE_TYPE := LEFT;
  DIVISOR  : Types.LONG_FIXED := RIGHT;
begin
  Machine_Dependent.Rate_Long_Div(DIVIDEND,DIVISOR,QUOTIENT);
  return QUOTIENT;
end "/";

function "*"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.LONG_FIXED is
  RESULT         : Types.LONG_FIXED;
  MULTIPLICAND_1 : Types.RATE_TYPE  := LEFT;
  MULTIPLICAND_2 : Types.LONG_FIXED := RIGHT;
begin
  Machine_Dependent.Rate_Long_Long_Mul(MULTIPLICAND_1,MULTIPLICAND_2,RESULT);
  return RESULT;
end "*";

function "*"(LEFT  : Types.RATE_TYPE;
             RIGHT : Types.LONG_FIXED) return Types.RATE_TYPE is
  RESULT         : Types.RATE_TYPE;
  MULTIPLICAND_1 : Types.RATE_TYPE  := LEFT;
  MULTIPLICAND_2 : Types.LONG_FIXED := RIGHT;
begin
  Machine_Dependent.Rate_Long_Rate_Mul(MULTIPLICAND_1,MULTIPLICAND_2,RESULT);
  return RESULT;
end "*";

function Power(BASE, RAISED_TO : Types.LONG_FIXED) return Types.LONG_FIXED is
  RESULT     : Types.LONG_FIXED := 1.0;
  OLD_RESULT : Types.LONG_FIXED;
begin
  for I in 1..Types.WORD(RAISED_TO) loop
    OLD_RESULT := RESULT;
    RESULT := RESULT * BASE;
    if RESULT = OLD_RESULT then             -- if no change, don't waste time
      exit;
    end if;
  end loop;
  return RESULT;
end Power;

function Sin (ANGLE : Types.BAM) return Types.LONG_FIXED is

--|
--| SUBPROGRAM BODY : Math.Sin
--|
--|   Sin is a function which takes an angle in Binary Angle
--| Measurements and uses a table lookup to find the corresponding result.
--| It returns the sin of the ANGLE in the Types.LONG_FIXED type.
--|
```

```
NEGAT  : Types.WORD := 1;
ANGLE2 : Types.WORD := Types.WORD(ANGLE);
TEMP   : Types.LONG_FIXED;

begin
  if ANGLE2 < 0 then
     ANGLE2 := abs(ANGLE2);
     NEGAT   := - 1;
  end if;
  if ANGLE2 > 16384 then
     TEMP := TRIG_FUNC(Types.WORD((32767 - ANGLE2)/163),SINE);
     return TEMP * Types.LONG_FIXED(NEGAT);
  else
     TEMP := TRIG_FUNC(Types.WORD(ANGLE2/163),SINE);
     return TEMP * Types.LONG_FIXED(NEGAT);
  end if;
end Sin;


function Cos (ANGLE : Types.BAM) return Types.LONG_FIXED is

--|
--| SUBPRCGRAM BODY : Math.Cos
--|
--|   Cos is a function which takes an angle in Binary Angle
--| Measurements and uses a table lookup to find the corresponding result.
--| The result is returned in the Types.LONG_FIXED type.
--|

ANGLE2 : Types.WORD := Types.WORD(ANGLE);

begin
  ANGLE2 := abs(ANGLE2);
  if ANGLE2 > 16384 then
     return (-1) * TRIG ,UNC(Types.WORD((32767-ANGLE2)/163),COSINE);
  else
     return TRIG_FUNC(Types.WORD(Angle2/163),COSINE);
  end if;
end Cos;

function Tan (ANGLE : Types.BAM) return Types.LONG_FIXED is

--|
--| SUBPROGRAM BODY : Math.Tan
--|
--|   Tan is the tangent function which takes an angle in Binary Angle
--| Measurements and uses a table lookup to find the corresponding result.
--|

  TANGENT     : Types.LONG_FIXED;
  THETA       : Types.WORD;
```

```
begin
  Time_Stamp.Log(0048);     --$TP(0048) Math.Tan start
  THETA := Types.WORD(ANGLE/182);      -- approx. 182 bams per degree
  if THETA >= -90 and THETA <= 90 then
    if THETA >= 0 then
      TANGENT := TAN_TABLE(THETA);
    else
      TANGENT := -TAN_TABLE(-THETA);
    end if;
  elsif THETA < -90 then
    TANGENT := TAN_TABLE(THETA + 180);
  else
    TANGENT := -TAN_TABLE(180-THETA);
  end if;
  Time_Stamp.Log(0049);     --$TP(0049) Math.Tan end
  return TANGENT;
end Tan;


function Sqrt(X : in Types.METERS) return Types.METERS is

--|
--| SUBPROGRAM BODY : Math.Sqrt
--|
--|   Sqrt returns the square root of a number.  This routine will exit when
--| the approximation of the square is close to the previous result.  This
--| prevents unneeded looping for accuracy.
--|

  use Types;   -- import operators
  F : Types.METERS := X;
  Y : Types.METERS := 1.0;
  OLD_Y : Types.METERS := Y;
begin
  Time_Stamp.Log(0050);     --$TP(0050) Math.Sqrt start (METERS)
  if X = 0.0 then
    return F;
  end if;
  for I in 1..15 loop
    exit when Y = 0.0;
    Y := ( Y + F/Y ) / Types.WORD(2);
    if Y = OLD_Y then
      exit;
    end if;
    OLD_Y := Y;
  end loop;
  Time_Stamp.Log(0051);     --$TP(0051) Math.Sqrt end (METERS)
  return Y;
end Sqrt;

function Sqrt(X : in Types.LONG_FIXED) return Types.LONG_FIXED is
```

```
  use Types;   -- import operators
  F : Types.LONG_FIXED := X;
  Y : Types.LONG_FIXED := 1.0;
  OLD_Y : Types.LONG_FIXED := Y;
begin
  Time_Stamp.Log(0052);      --$TP(0052) Math.Sqrt start (LONG_FIXED)
  if X = 0.0 then
    return F;
  end if;
  for I in 1..15 loop
    exit when Y = 0.0;
    Y := ( Y + F/Y ) / Types.WORD(2);
    if Y = OLD_Y then
      exit;
    end if;
    OLD_Y := Y;
  end loop;
  Time_Stamp.Log(0053);      --$TP(0053) Math.Sqrt end (LONG_FIXED)
  return Y;
exception
  when NUMERIC_ERROR => Y := OLD_Y;
                        return Y;
end Sqrt;

--
--  for RATE TYPE
--
function Sqrt(X : in Types.RATE_TYPE) return Types.RATE_TYPE is
  use Types;   -- import operators
  F : Types.RATE_TYPE := X;
  Y : Types.RATE_TYPE := 1.0;
  OLD_Y : Types.RATE_TYPE := Y;
begin
  Time_Stamp.Log(0116);      --$TP(0116) Math.Sqrt start (RATE_TYPE)
  if X = 0.0 then
    return F;
  end if;
  for I in 1..15 loop
    exit when Y = 0.0;
    Y := ( Y + F/Y ) / Types.WORD(2);
    if Y = OLD_Y then
      exit;
    end if;
    OLD_Y := Y;
  end loop;
  Time_Stamp.Log(0117);      --$TP(0117) Math.Sqrt end (RATE_TYPE)
  return Y;
exception
  when NUMERIC_ERROR => Y := OLD_Y;
                        return Y;
end Sqrt;
```

```
--|
--| SUBPROGRAM BODY : Math.Arcsin
--|
--|   Arcsin uses an appoximation method of series expansion in order to achieve
--| it's results.  Simple constants are named to improve readability.  The
--| constants are not fully divided out beforehand to increase accuracy, but
--| to improve speed, the constants could be divided beforehand and constants
--| used to multiply the TERMs.
--|

function Arcsin(THETA : Types.LONG_FIXED) return Types.LONG_FIXED is
  rad_to_deg : constant Types.LONG_FIXED := 57.296875;  -- 180 / pi
  three      : constant Types.LONG_FIXED := 3.0;
  six        : constant Types.LONG_FIXED := 6.0;
  five       : constant Types.LONG_FIXED := 5.0;
  seven      : constant Types.LONG_FIXED := 7.0;
  fifteen    : constant Types.LONG_FIXED := 15.0;
  forty      : constant Types.LONG_FIXED := 40.0;
  three_thirty_six : constant Types.LONG_FIXED := 336.0;

  TERM1      : Types.LONG_FIXED;
  TERM2      : Types.LONG_FIXED;
  TERM3      : Types.LONG_FIXED;
  TERM4      : Types.LONG_FIXED;
  RESULT     : Types.LONG_FIXED;

begin
  TERM1 := THETA;
  TERM2 := Power(THETA,three);
  TERM2 := TERM2 / six;
  TERM3 := three * Power(THETA,five);
  TERM3 := TERM3 / forty;
  TERM4 := fifteen * Power(THETA,seven);
  TERM4 := TERM4 / three_thirty_six;
  RESULT := (TERM1 + TERM2 + TERM3 + TERM4) * rad_to_deg;
  return RESULT;
end Arcsin;

function Arctan(Z_INPUT : Types.LONG_FIXED) return Types.BAM is

 --|
 --| SUBPROGRAM BODY : Math.Arctan
 --|
 --| A function used to return an approximation of the arctangent function.
 --| Using the Taylor series expansion:
 --| arctan z = z - (z**3/3) + (z**5/5) - (z**7/7) + ... (|z|<= 1 and z**2 /= -1)
 --| carried out for two terms (initially).
 --|

 Z_CUBED       : Types.LONG_FIXED;
```

```
QUOTIENT       : Types.LONG_FIXED;
ARCTAN_Z       : Types.LONG_FIXED;
CONV_FACTOR    : constant Types.LONG_FIXED := 10430.38;  --Radians to BAMS
TEMP           : Types.LONG_FIXED;
ARCTAN_Z_BAMS : Types.BAM;

begin
  Z_CUBED := Z_INPUT * Z_INPUT;  --actually z**2
  Z_CUBED := Z_CUBED * Z_INPUT;  -- z**3
  QUOTIENT := Z_CUBED / Types.WORD(3);
  ARCTAN_Z := Z_INPUT - QUOTIENT;
  TEMP := ARCTAN_Z * CONV_FACTOR;
  ARCTAN_Z_BAMS := Types.BAM(TEMP);
  return ARCTAN_Z_BAMS;
end Arctan;

function Get_Random_Num(LIMIT : Types.METERS) return Types.METERS is

--|
--| SUBPROGRAM BODY : Math.Get_Random_Num
--|
--|  This function returns a psuedo random number to the caller.  It has
--| three forms returning three different types for the convenience of
--| Simulate.Sensor.  The random number is received from the channel two
--| counter.  Therefore the number returned from the
--| Machine_Dependent.Next_Random call is between 0 and Machine_Dependent
--| max_timer_value.  This number is then multiplied by the parameter
--| LIMIT, and, returned in the type of LIMIT, is a random number from
--| 0 to LIMIT.
--|

  VALUE : Types.RATE_TYPE;
  TEMP  : Types.WORD_INDEX;
begin
  Machine_Dependent.Next_Random(TEMP);
  VALUE := Types.RATE_TYPE(TEMP) / Types.WORD(
                                   Machine_Dependent.max_timer_value);
  VALUE := VALUE * Types.RATE_TYPE(LIMIT);
  return Types.METERS(VALUE);
end Get_Random_Num;

function Get_Random_Num(LIMIT : Types.LONG_FIXED) return Types.LONG_FIXED is
  VALUE : Types.RATE_TYPE;
  TEMP  : Types.WORD_INDEX;
begin
  Machine_Dependent.Next_Random(TEMP);
  VALUE := Types.RATE_TYPE(TEMP) / Types.WORD(
                                   Machine_Dependent.max_timer_value);
  VALUE := VALUE * LIMIT;
  return Types.LONG_FIXED(VALUE);
end Get_Random_Num;
```

```
function Get_Random_Num(LIMIT : Types.WORD_INDEX) return Types.WORD_INDEX is
  VALUE : Types.RATE_TYPE;
  TEMP  : Types.WORD_INDEX;
begir
  Machine_Dependent.Next_Random(TEMP);
  VALUE := Types.RATE_TYPE(TEMP) / Types.WORD(
                                    Machine_Dependent.max_timer_value);
  VALUE := VALUE * Types.RATE_TYPE(LIMIT);
  return Types.WORD_INDEX(VALUE);
end Get_Random_Num;

end Math;
```

```
---------------------------------------------------------------------
--% UNIT:      Mouse Package Spec.                                  --
--% Effects:  Provides graphics pointing device interrupt handling.    --
--% Modifies: Status Mode, and Mouse_Buffer X-Y positions are updated. --
--% Requires: Runtime initialization of interrupt vector.              --
--% Raises:   Task will terminate on MOUSE_ERROR.                      --
--% Engineer: M. Sperry.                                               --
---------------------------------------------------------------------

--|
--| PACKAGE SPEC : Mouse
--|
--|   In addition to establishing communications with the mouse, a task is
--| provided which handles the receive interrupt generated by the mouse at
--| COM2.  This task has the pragma INTERRUPT_HANDLER and special restrictions
--| apply to it's communication facilities in order to guarantee a good response
--| time.
--|

-- Modifications Log
--
-- 88-09-30 : MPS => Original created.
--

with System;

package Mouse is


procedure Initialize;

task Char_In is
  pragma INTERRUPT_HANDLER;
  entry REPORT;
  for REPORT use at (16#83#,0);                  -- COM2 8250 serial port vector
end Char_In;

end Mouse;
```

```
------------------------------------------------------------------
--% UNIT:     Mouse Package Body.                                --
--% Effects:  Provides graphics pointing device interrupt handling.   --
--% Modifies: Status Mode, and Mouse_Buffer X-Y positions are updated.  --
--% Requires: Runtime initialization of interrupt vector.        --
--% Raises:   Task will terminate on MOUSE_ERROR.                --
--% Engineer: M. Sperry.                                         --
------------------------------------------------------------------

--|
--| PACKAGE BODY : Mouse
--|
--|   The Mouse package implements the routines needed for control of a
--| mouse.  There is an initialization procedure which sets up the mouse
--| for 4800 baud, no parity, 7 data bits, and two stop bits.  There is
--| also an interrupt entry task which takes data from the mouse and
--| if a complete report is generated, gives that data to the Mouse_Buffer
--| task.
--|

-- Modifications Log
--
-- 88-09-30 : MPS => Original created.
-- 89-08-08 : MPS => Made all references to hardware in one package.
-- 89-12-06 : TEG => Shut off mouse interrupts during init.  Added support
--                   for Restarting the system using RESET&MODE buttons.
--

with Types;
with Low_Level_IO;
with Debug_IO;
with Mouse_Buffer;
with Mouse_Data;                 -- provides constants and data structures
with Status;
with Interrupt_Control;
with Time_Stamp;
with HW_Config;
with Distrib;
use Low_Level_IO;
use Mouse_Data;                  -- visibility to "and" function

pragma ELABORATE(Low_Level_IO, Debug_IO, Mouse_Buffer, Status, Time_Stamp);

package body Mouse is

DATA              : Low_Level_IO.BYTE;            -- char from mouse
BUTTON_PUSHED     : Mouse_Data.BIT_FIELD;         -- array representing keys
STATUS_BYTE       : Mouse_Data.BIT_FIELD;         -- represents status errors
PREV_BUTTON_PUSH : Mouse_Data.BIT_FIELD := (others => FALSE); --previous buttons
MOUSE_INPUT       : Mouse_Data.RAW_MOUSE_WORD := (0,0,0);-- transform to 12-bit
```

```
MOUSE_REPORT      : Mouse_Data.SIGNED_MOUSE_WORD; -- transformation to signed
REPORT_COUNT      : Types.WORD range 0..5 := 0;   -- counts byte in report
CHANGE_REQUESTED : BOOLEAN := FALSE;             -- rendezvous with status?
MOUSE_ERROR       : EXCEPTION;

TEMP_X            : Types.WORD;                   -- local copy of X motion
TEMP_Y            : Types.WORD;                   -- local copy of Y motion

procedure Initialize is

--| SUBPROGRAM BODY : Mouse.Initialize
--|
--|   Initialize sets up the mouse at 4800 baud, no parity, 7 data bits, and
--| two stop bits.  The number of stop bits is insignificant.  There should
--| only be two formats that the mouse can be in, either relative bit pad one
--| or Micrsoft Mouse.  The default on power up for the mouse is MM at 4800.
--| The mouse must be commanded in the following order: BAUD (which is set to
--| default to 4800 so it is not necessary to reprogram it), # of reports/sec.,
--| and then format of the reports.  The mouse used is a Logitech Serial Mouse
--| as described in hwconfig.as.  The mouse is programmed with Relative Bit
--| Pad One format which has five bytes of data associated with it.
--|

  INTERRUPTS : Low_Level_IO.BYTE;              -- for input of 8259 ints
  RESPONSE   : Low_Level_IO.BYTE;              -- for mouse responses
  TIME_OUT   : INTEGER := 30000;               -- time out for mouse response

begin
--  Disable receive interrupts
  Send_Control(HW_Config.COM2_int_enable,Mouse_Data.specific_int_disable);
  Receive_Control(HW_Config.COM2_status,RESPONSE);  -- clean out junk in status
  Receive_Control(HW_Config.COM2_data,RESPONSE);     -- clean out junk in data
  Send_Control(HW_Config.COM2_control,Mouse_Data.access_baud);
  Send_Control(HW_Config.COM2_data,Mouse_Data.host_baud); -- set BAUD = 4800
-- set COM2 serial parameters
  Send_Control(HW_Config.COM2_control,Mouse_Data.host_format);
  Send_Control(HW_Config.COM2_data,Mouse_Data.acknowledge); -- wakeup mouse
  loop
    Receive_Control(HW_Config.COM2_status,RESPONSE);    -- wait for response
    if RESPONSE = Mouse_Data.data_new then
      Receive_Control(HW_Config.COM2_data,RESPONSE);    -- clear out byte
      exit;
    else
      TIME_OUT := TIME_OUT - 1;
    end if;
    if TIME_OUT = 0 then
      exit;
    end if;
  end loop;
  if TIME_OUT = 0 then
    Debug_IO.Put_Line("Unable to establish communications with mouse.");
```

```
  end if;
  Send_Contro\(HW_Config.COM2_data,Mouse_Data.mouse_char_speed);
  delay 0.01;                                     -- slow for mouse input buffer
  Send_Control(HW_Config.COM2_data,Mouse_Data.mouse_format);
  Send_Control(HW_Config.COM2_modem_control,Mouse_Data.general_int_enable);
  Send_Control(HW_Config.COM2_int_enable,Mouse_Data.specific_int_enable);
  Receive_Control(HW_Config.pic_8259_mr,INTERRUPTS);
-- enable COM2 in PIC in line below
  INTERRUPTS := Mouse_Data.Bits_to_Byte
           (Mouse_Data.Byte_to_Bits(INTERRUPTS) and Mouse_Data.pic_and_mask);
  Send_Control(HW_Config.pic_8259_mr,INTERRUPTS);
end Initialize;


task body Char_In is

--|
--| TASK BODY : Mouse.Char_In
--|
--|  One of the main tasks used to move the reticle around the battlefield
--| screen.  The task rendezvous with the graphics task reporting positions
--| every 28 milliseconds, unless the middle button is pressed (MODE) changing
--| the mode to AUTOMATIC.  In this event, the mouse simply waits for a change
--| to MANUAL, since automatic mode is controlled by the rocket task.  The mouse
--| task will not rendezvous with the graphics task until set to MANUAL.  When
--| in MANUAL mode, the task (upon completion of one report) will rendezvous
--| with the graphics task at high priority to report it's position.  It will
--| then change the status task's shared variables if any need to be changed.
--| If one does, and the status task has completed it's previous work and gone
--| to an accept state, then the mouse task wakes it up.  Because the mouse
--| is programmed with Relative Bit Pad One format and needs five bytes of data
--| in order to complete its report, after the first byte has come in, it is
--| only 2 milliseconds until the next byte comes in until five bytes have been
--| received.  Then there is a gap of 18 milliseconds until the next byte will
--| be seen (assuming constant motion of the mouse).  This is why there is
--| very little processing of data until the fifth byte. It is entirely possible
--| because the mouse is an asychronous device that up to three reports may be
--| generated and handled in one interval.  This worst case must be accounted
--| for in timing considerations.
--|

use Status;                                     -- for visibility to "="
use Types;                                      -- for visibility to "+"

begin
  loop
    accept Report do
      --$TP(0056) Mouse task start
      Receive_Control(HW_Config.COM2_status,DATA); -- receive status
      STATUS_BYTE := Mouse_Data.Byte_to_Bits(DATA);--check statusbyte for errors
      if STATUS_BYTE(Mouse_Data.overflow) or
```

```
    STATUS_BYTE(Mouse_Data.framing) then
   REPORT_COUNT := 0;                          -- start a new report
   Receive_Control(HW_Config.COM2_data,DATA); -- clear out data port
 else
   Receive_Control(HW_Config.COM2_data,DATA); -- get valid data
   if DATA > Mouse_Data.sync_byte then         -- check for new report
     REPORT_COUNT := 1;                      -- start of new report
   end if;
 end if;
 case REPORT_COUNT is                        -- convert data to mouse X,Y
   when 1 =>                                 -- or buttons.
     BUTTON_PUSHED := Mouse_Data.Byte_to_Bits(DATA);
     REPORT_COUNT := REPORT_COUNT + 1;
   when 2 =>
     MOUSE_INPUT.LOW := Mouse_Data.Byte_to_Bit6(DATA);
     REPORT_COUNT := REPORT_COUNT + 1;
   when 3 =>
     MOUSE_INPUT.HIGH := Mouse_Data.Byte_to_Bit6(DATA);
     MOUSE_REPORT := Mouse_Data.Raw_to_Signed(MOUSE_INPUT);
     TEMP_X := MOUSE_REPORT.LOW12;
     REPORT_COUNT := REPORT_COUNT + 1;
   when 4 =>
     MOUSE_INPUT.LOW := Mouse_Data.Byte_to_Bit6(DATA);
     REPORT_COUNT := REPORT_COUNT + 1;
   when 5 =>
     -- don't move mouse if any buttons pushed.
     if (not BUTTON_PUSHED(Mouse_Data.reset)) and -- guarantee only one -
        (not BUTTON_PUSHED(Mouse_Data.mode)) and-- rendezvous per report
        (not BUTTON_PUSHED(Mouse_Data.launch)) then  -- (RTE bug) -
       PREV_BUTTON_PUSH(Mouse_Data.reset)  := FALSE;
       PREV_BUTTON_PUSH(Mouse_Data.mode)   := FALSE;
       PREV_BUTTON_PUSH(Mouse_Data.launch) := FALSE;
       MOUSE_INPUT.HIGH := Mouse_Data.Byte_to_Bit6(DATA);
       MOUSE_REPORT := Mouse_Data.Raw_to_Signed(MOUSE_INPUT);
       TEMP_Y := MOUSE_REPORT.LOW12;
       if Status.MODE = Status.MANUAL then
         MOUSE_BUFFER.MOUSE_X := TEMP_X;
         MOUSE_BUFFER.MOUSE_Y := TEMP_Y;
         --$TP(0057) Mouse rendezvous with Save start
       select    -- must be conditional to work in INTERRUPT_HANDLER
           Mouse_Buffer.Save.Reticle_Motion;
           --$TP(0058) Mouse rendezvous with Save end
         else
           null;
         end select;
       end if;
     else
--
--  A BUTTON IS DEPRESSED.  FIRST LOOK AT "RESET" BUTTON
--
       if BUTTON_PUSHED(Mouse_Data.reset) and
```

```
             not PREV_BUTTON_PUSH(Mouse_Data.reset) then
            for I in Status.RESET_STATUS_TYPE loop
              Status.STATUS_CONTROL(I).DATA := 0;
              Status.STATUS_CONTROL(I).DISPLAYED := FALSE;
            end loop;
            Status.REQ_COUNT := Status.REQ_COUNT + 1;
            CHANGE_REQUESTED := TRUE;
            PREV_BUTTON_PUSH(Mouse_Data.reset) := TRUE;
          else
            PREV_BUTTON_PUSH(Mouse_Data.reset) := FALSE;
          end if;
--
-- NOW LOOK AT MODE BUTTON...
-- When the MODE button is pushed, check to see if the RESET button is
-- currently active.  If so, then do a system reset!
--
          if BUTTON_PUSHED(Mouse_Data.mode) then
            if BUTTON_PUSHED(Mouse_Data.reset) then
              Distrib.Restart;                   -- perform system shutdown
            elsif not PREV_BUTTON_PUSH(Mouse_Data.mode) then
              if Status.MODE = Status.MANUAL then  -- Change mode
                Status.MODE := Status.AUTOMATIC;
              else
                Status.MODE := Status.MANUAL;
              end if;
              Status.MODE_DISPLAYED := FALSE;
              Status.REQ_COUNT := Status.REQ_COUNT + 1;
              CHANGE_REQUESTED := TRUE;
              PREV_BUTTON_PUSH(Mouse_Data.mode) := TRUE;
            end if;
          else
            PREV_BUTTON_PUSH(Mouse_Data.mode) := FALSE;
          end if;
--
-- FINALLY, LOOK AT LAUNCH BUTTON
--
          if BUTTON_PUSHED(Mouse_Data.launch) and
             not PREV_BUTTON_PUSH(Mouse_Data.launch) then
            if Status.MODE = Status.MANUAL then
              Mouse_Buffer.LAUNCH := TRUE;
              Mouse_Buffer.NEW_ABS_X := Mouse_Buffer.OLD_ABS_X;
              Mouse_Buffer.NEW_ABS_Y := Mouse_Buffer.OLD_ABS_Y;
            end if;
            PREV_BUTTON_PUSH(Mouse_Data.launch) := TRUE;
          else
            if not BUTTON_PUSHED(Mouse_Data.launch) then
              PREV_BUTTON_PUSH(Mouse_Data.launch) := FALSE;
            end if;
          end if;
          if CHANGE_REQUESTED and then Status.REQ_COUNT = 1 then
            --$TP(0059) Mouse rendezvous with Status start
```

```
            select
              Status.Update.Signal;
              --$TP(0060) Mouse rendezvous with Status end
            else
              null;
            end select;
          end if;
        end if;
        CHANGE_REQUESTED := FALSE;
        REPORT_COUNT := 0;
      when others => null;
    end case;
    Send_Control(HW_Config.pic_8259,Mouse_Data.spec_eoi); -- specific EoI
    --$TP(0061) Mouse task end
   end Report;
  end loop;
end Char_In;

end Mouse;
```

```
------------------------------------------------------------------
--% UNIT:     Mouse_Buffer Package Spec.                          --
--% Effects:  Buffers mouse data input, translates it to pixel system.  --
--% Modifies: No global data is modified (other than in own spec).     --
--% Requires: No initialization is required.                      --
--% Raises:   No explicitly raised exceptions are propagated.     --
--% Engineer: M. Sperry.                                          --
------------------------------------------------------------------

--|
--| PACKAGE SPEC : Mouse_Buffer
--|
--|   Package Mouse_Buffer contains a task called Save which is responsible
--| for saving reports of mouse movement via a rendezvous with an interrupt
--| task.  The task then rendezvous with the display task to relocate the
--| reticle.  The shared variables of the X and Y positions as well as the
--| launch flag are contained here.  The mode flag is contained in the
--| status package specification.
--|

-- Modifications Log
--
-- 88-10-24 : MPS => Original created.
--

with Types;
with Config;

package Mouse_Buffer is

stack_size : constant := 118;  -- in bytes

MOUSE_X     : Types.WORD;     -- for use with the Save task in Mouse_Buffer
MOUSE_Y     : Types.WORD;     -- for use with the Save task in Mouse_Buffer
LAUNCH      : BOOLEAN := FALSE;

OLD_ABS_X   : Types.WORD;     -- absolute X position of Reticle on Screen
OLD_ABS_Y   : Types.WORD;     --      "    Y    "        "       "
NEW_ABS_X   : Types.WORD;     -- for use by ENGAGE (latched values by Mouse pkg)
NEW_ABS_Y   : Types.WORD;

task type Save_Type is
  entry Reticle_Motion;
  pragma PRIORITY(Config.save_priority);
end Save_Type;
for Save_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                                          stack_size);
Save        : Save_Type;      -- for saving motion of mouse to display

end Mouse_Buffer;
```

```
---------------------------------------------------------------------------
--% UNIT:     Mouse_Buffer Package Body.                                  --
--% Effects:  Buffers mouse data input, translates it to pixel system.    --
--% Modifies: No global data is modified (other than in own spec).        --
--% Requires: No initialization is required.                              --
--% Raises:   No explicitly raised exceptions are propagated.             --
--% Engineer: M. Sperry.                                                  --
---------------------------------------------------------------------------

--|
--| PACKAGE BODY : Mouse_Buffer
--|
--|   Package body Mouse_Buffer is responsible for the implementation of the
--| buffering between the mouse interrupt routine and the screen.  Note that
--| checks are performed to be sure that the reticle is within the screen
--| defined by Config.  Also, note that the Y coordinate is reversed because
--| the screen on the EGA runs (in the Y direction) from 0 to 349 starting
--| from the upper left and moving down, i.e., the mouse has Y direction as
--| positive moving up, and the EGA has Y positive moving down.
--|

-- Modifications Log
--
-- 88-10-24 : MPS => Original created.
--

with Shapes;
with Graphics;
with Config;
with Debug_IO;
with Interrupt_Control;
with Time_Stamp;
pragma ELABORATE(Debug_IO, Graphics, Interrupt_Control, Time_Stamp);

package body Mouse_Buffer is
use Types;                                  -- neeued for visibility to '+'


task body Save_Type is

list_len     : constant := 1;

left_limit   : constant := Config.battlefield_screen_left;
right_limit  : constant := Config.battlefield_screen_right;
top_limit    : constant := Config.battlefield_screen_top;
bottom_limit : constant := Config.battlefield_screen_bottom;

PRIORITY     : Graphics.PRIORITY_TYPE := Graphics.HIGH;
WORK_LIST    : Graphics.MOVE_LIST_TYPE(list_len .. list_len);-- 1 item (reticle)

TEMP_X       : Types.WORD;
```

```
TEMP_Y        : Types.WORD;

begin
--
-- Initial display of reticle
--
  WORK_LIST(list_len).XY_OLD := (Config.battlefield_center_x, Config.battlefield_center_y);
  WORK_LIST(list_len).XY_NEW := (Config.battlefield_center_x, Config.battlefield_center_y);
  WORK_LIST(list_len).OBJECT := Shapes.RETICLE;
  WORK_LIST(list_len).COLOR := Graphics.reticle_color;

  Graphics.Display.Move(PRIORITY, WORK_LIST);

  loop
    begin                    -- exception block
      Time_Stamp.Log(0062);     --$TP(0062) Mouse Buffer task and accept
      accept Reticle_Motion;
--
-- Get new positions of reticle (mouse)
--
      Interrupt_Control.Disable;
      TEMP_X := WORK_LIST(list_len).XY_OLD.X + MOUSE_X;
      TEMP_Y := WORK_LIST(list_len).XY_OLD.Y - MOUSE_Y;
      Interrupt_Control.Enable;
--
--   Check bounds of reticle; don't let it go past edge of battlefied screen.
--
      if (TEMP_X + Shapes.RETICLE_LEFT) < left_limit then
        TEMP_X := left_limit - Shapes.RETICLE_LEFT;
      elsif (TEMP_X + Shapes.RETICLE_RIGHT) > right_limit then
        TEMP_X := right_limit - Shapes.RETICLE_RIGHT;
      end if;
      if (TEMP_Y + Shapes.RETICLE_TOP) < top_limit then
        TEMP_Y := top_limit - shapes.reticle_top;
      elsif (TEMP_Y + Shapes.RETICLE_BOTTOM) > bottom_limit then
        TEMP_Y := bottom_limit - Shapes.RETICLE_BOTTOM;
      end if;

      WORK_LIST(list_len).XY_NEW.X := TEMP_X;
      WORK_LIST(list_len).XY_NEW.Y := TEMP_Y;


--
--  update global accessable values
--
      Interrupt_Control.Disable;
      OLD_ABS_X := TEMP_X;
      OLD_ABS_Y := TEMP_Y;
      Interrupt_Control.Enable;

      Time_Stamp.Log(0063);     --$TP(0063) Mouse_Buffer rendezvous with Graphics start
      Graphics.Display.Move(PRIORITY, WORK_LIST);
```

```
      Time_Stamp.Log(0064);      --$TP(0064) Mouse_Buffer rendezvous with Graphics end

      WORK_LIST(list_len).XY_OLD := WORK_LIST(list_len).XY_NEW;
    exception
      when others =>
        Debug_IO.Put_Line("Error in Save");
    end;                                       -- exception block
  Time_Stamp.Log(0065);      --$TP(0065) Mouse_Buffer task end
  end loop;

end Save_Type;

end Mouse_Buffer;
```

```
------------------------------------------------------------------
--% UNIT:      Mouse_Data Specification.                        --
--% Effects:   Provides relevant data structures and constants. --
--% Modifies:  Nothing.                                         --
--% Requires:  Nothing.                                         --
--% Raises:    Nothing.                                         --
--% Engineer:  M. Sperry.                                       --
------------------------------------------------------------------


--|
--| PACKAGE SPEC : Mouse_Data
--|
--|   Package Mouse_Data provides the data structures and constants necessary
--| to intialize and run a Logitech C7 serial mouse at 4800 baud, no parity, 7
--| data bits, and two stop bits.  The Logitech mouse is capable of 8 different
--| formats.  Relative Bit Pad One is chosen here because it allows twelve
--| bits of motion data for each repo t.  Although this creates more work in
--| the processing of each byte of data (there are five bytes of data in each
--| report) there is more accuracy in the pointing device.
--|   The mouse controls the movement of the reticle (defined in the Graphics
--| package) by receiving a report, generating the motion in X and Y
--| coordinates, and sending these values to package Mouse_Buffer for
--| processing (task Save).
--|   The reports come in 28 msecs apart with a 2 msec interval between each
--| byte of the report.  A report consists of the following in Relative Bit Pad
--| One:
--|
--| P    6    5    4    3    2    1    0    = bit number
--| ----------------------------------------------------
--| np   1    0    L    M    R    0    0    Byte 1  (minimum value = 64)
--| np   0    X5   X4   X3   X2   X1   X0   Byte 2
--| np   0    X11  X10  X9   X8   X7   X6   Byte 3
--| np   0    Y5   Y4   Y3   Y2   Y1   Y0   Byte 4
--| np   0    Y11  Y10  Y9   Y8   Y7   Y6   Byte 5
--|  \- no parity
--| L,M,R above stands for Left, Middle, and Right buttons; 1 = key pressed.
--|
--|   The mouse is located at COM2 on an AT which is base address 2F8 (hex).
--|

-- Modifications Log
--
-- 89-04-15 : MPS => Original created.
-- 89-08-08 : MPS => Defined COM2 addresses in HW_Config.
--

with Types;
with Low_Level_IO;
with Unchecked_Conversion;
with HW_Config;
use Low_Level_IO;
```

```
package Mouse_Data is

spec_eoi          : constant Low_Level_IO.BYTE := 16#63#;--specific end int
sync_byte         : constant Low_Level_IO.BYTE := 63; -- used to sync reports

-- The following constants are bit masks to be used with the BIT_FIELD type.

reset             : constant := 4;     -- left button (reset statistics)
mode              : constant := 3;     -- middle button (change mode)
launch            : constant := 2;     -- right button (fire rocket)

-- These constants are declared to aid in detecting serial errors during
-- transmision.

overflow          : constant := 1;     -- position from status (2FD)
framing           : constant := 2;     -- position from status (2FD)

-- Because the data bits are received six bits at a time, the following record
-- representation clauses are used to convert two bytes of data (a least and
-- most significant) to a single signed twelve bit number.

type BIT6_TYPE is range 0 .. 63;
type GAP_TYPE is range 0 .. 15;

type RAW_MOUSE_WORD is
  record
    LOW  : BIT6_TYPE;
    HIGH : BIT6_TYPE;
    GAP  : GAP_TYPE;
  end record;

for RAW_MOUSE_WORD use
  record
    LOW  at 0 range 0 .. 5;
    HIGH at 0 range 6 .. 11;
    GAP  at 0 range 12 .. 15;
  end record;

least_low12 : constant := -2048;

type SIGNED_MOUSE_WORD is
  record
    LOW12 : Types.WORD range least_low12 .. 2047;
    GAP   : Types.WORD range 0 .. 15;
  end record;

for SIGNED_MOUSE_WORD use
  record
    LOW12 at 0 range 0 .. 11;
    GAP   at 0 range 12 .. 15;
```

```
  end record;

-- Most significant bit for the following type definition on TANDY 4000 : 15

type BIT_FIELD is array(0..15) of BOOLEAN;
pragma pack(BIT_FIELD);
for BIT_FIELD'size use 16;

function Raw_to_Signed is new Unchecked_Conversion(RAW_MOUSE_WORD,
                                                    SIGNED_MOUSE_WORD);
function Byte_to_Bit6 is new Unchecked_Conversion(Low_Level_IO.BYTE,BIT6_TYPE);
function Bits_to_Byte is new Unchecked_Conversion(BIT_FIELD,Low_Level_IO.BYTE);
function Byte_to_Bits is new Unchecked_Conversion(Low_Level_IO.BYTE,BIT_FIELD);

pic_and_mask     : constant BIT_FIELD :=
        (TRUE,TRUE,TRUE,FALSE,TRUE,TRUE,TRUE,TRUE,TRUE,
         TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE);     -- will enable level 03 (COM2)

-- The following constants are used in the initialization procedure of mouse.
-- They are used to access the serial port COM2 on a TANDY PC (386).

access_baud         : constant Low_Level_IO.BYTE := 16#80#;
-- access baud rate rgtrs.
host_baud           : constant Low_Level_IO.BYTE := 16#18#;
-- 4800 baud (30h = 2400)
host_format         : constant Low_Level_IO.BYTE := 16#1E#;
-- 4800,e,7,1
acknowledge         : constant Low_Level_IO.BYTE := 16#20#;
-- mouse responds w/06h when
ack_response        : constant Low_Level_IO.BYTE := 16#06#;
-- sent a space (20h)
data_new            : constant Low_Level_IO.BYTE := 16#61#;
-- char received
mouse_format        : constant Low_Level_IO.BYTE := 16#42#;
-- Relative Bit Pad One
mouse_char_speed    : constant Low_Level_IO.BYTE := 16#4C#;
-- 35 reports/sec when moving
general_int_enable  : constant Low_Level_IO.BYTE := 16#08#;
-- for modem control register
specific_int_disable : constant Low_Level_IO.BYTE := 16#00#;
-- disable receive interrupt
specific_int_enable : constant Low_Level_IO.BYTE := 16#01#;
-- enable receive interrupt
pic_8259_mr         : constant Low_Level_IO.PORT_ADDRESS := 16#21#;

end Mouse_Data;
```

```
----------------------------------------------------------------------
--% UNIT:     Parameter Data Base Spec.                              --
--% Effects:  Provides rocket data types and initial values.         --
--% Modifies: No global data is modified.                            --
--% Requires: No initialization is required.                         --
--% Raises:   No exceptions.                                         --
--% Engineer: R. Chevier                                             --
----------------------------------------------------------------------

--|
--| PACKAGE SPEC : Parameter_Data_Base
--|
--|   This package defines the necessary default values for the rocket and
--| the targets.  There are four different target types described in
--| Simulate.Sensor.Targ_Sup.  The type of rocket used to attack these targets
--| is described by the values below.
--|

with Types;

package Parameter_Data_Base is

------------------------------------------
--                                      --
--       R O C K E T   V A L U E S      --
--                                      --
------------------------------------------
--  type DEGREES_TYPE is digits 6 range 0.0..360.0;
--  type RATE_TYPE is digits 5;
--  subtype MAX_ROCKET_RANGE is Types.WORD range 1..100;
--  MAX_DEGRADED_ROCKETS : MAX_ROCKET_RANGE := 1;
--  type MASS_TYPE is digits 5 range 10.0..100.0;
--  type THRUST_TYPE is digits 6 range 100.0..100000.0;
--  type BURN_RATE_TYPE is digits 5 range 0.001..10.0;
--  type RESISTANCE_TYPE is digits 5 range 0.001..100.0;
--  type DRIFT_VELOCITY_TYPE is digits 5 range 0.0..0.5;
--  subtype ROCKET_TURN_ACCEL_TYPE is RATE_TYPE range 0.01..1000.0;

  c_mass             : constant := 40.0;      -- kgs
  c_fuel             : constant := 300.0;     -- kgs
  c_thrust           : constant := 6000.0;    -- Newtons
  c_burn_rate        : constant := 5.0;       -- kgs/sec
  c_turn_burn_rate   : constant := 0.05;      -- kgs/degree
  c_forward_drag     : constant := 0.1875;    -- Newton-secs/meter (was 0.09375)
  c_side_drag        : constant := 0.203125;  -- Newton-secs/meter
  c_drift            : constant := 0.0;       -- meters/sec
  c_turn_rate        : constant := 200.0;     -- degrees/sec

  type ROCKET_PARAMETER_TYPE is record
    MASS             : Types.LONG_FIXED := c_mass;
    FUEL             : Types.LONG_FIXED := c_fuel;
```

```
    THRUST           : Types.LONG_FIXED := c_thrust;
    BURN_RATE        : Types.LONG_FIXED := c_burn_rate;
    TURN_BURN_RATE   : Types.LONG_FIXED := c_turn_burn_rate;
    FORWARD_DRAG     : Types.LONG_FIXED := c_forward_drag;
    SIDE_DRAG        : Types.LONG_FIXED := c_side_drag;
    DRIFT            : Types.LONG_FIXED := c_drift;
    TURN_RATE        : Types.LONG_FIXED := c_turn_rate;
  end record;

----------------------------------------
--                                    --
--      T A R G E T   V A L U E S     --
--                                    --
----------------------------------------
  type TARGET_PARAMETER_TYPE is record
    MAX_VELOCITY_Y    : Types.METERS; -- maximum velocity in Y per interval
    MAX_VELOCITY_X    : Types.METERS; -- maximum velocity in x per interval
    DELTA_VELOCITY_X : Types.METERS; -- maximum change in x per interval
    CHANGE_DIR_FREQ   : Types.WORD_INDEX; -- freq that x dir changes in intrvls
  end record;

  type TARGET_PARAMS_ARRAY is array(Types.TARGET_CLASS_TYPE) of
                                                 TARGET_PARAMETER_TYPE;
  TARGET_PARAMS : TARGET_PARAMS_ARRAY :=
                             ( Types.UNKNOWN  =>
                                 ( MAX_VELOCITY_Y    => 2.000,
                                   MAX_VELOCITY_X    => 1.500,
                                   DELTA_VELOCITY_X => 0.125,
                                   CHANGE_DIR_FREQ   => 25),
                               Types.T80      =>
                                 ( MAX_VELOCITY_Y    => 1.750,
                                   MAX_VELOCITY_X    => 1.250,
                                   DELTA_VELOCITY_X => 0.125,
                                   CHANGE_DIR_FREQ   => 21),
                               Types.SA9      =>
                                 ( MAX_VELOCITY_Y    => 1.875,
                                   MAX_VELOCITY_X    => 1.375,
                                   DELTA_VELOCITY_X => 0.125,
                                   CHANGE_DIR_FREQ   => 23),
                               Types.BMP2     =>
                                 ( MAX_VELOCITY_Y    => 1.250,
                                   MAX_VELOCITY_X    => 0.875,
                                   DELTA_VELOCITY_X => 0.125,
                                   CHANGE_DIR_FREQ   => 15));
--
-- To simplify, all target types currently have the same DELTA_VELOCITY_X
-- (meaning the same acceleration) and the same pixel size representation.
-- Therefore, they all have the same right and left border limits, and
-- consequently are all created within these borders.
--
  right_border_limit : constant := 3940.0;
```

```
  left_border_limit  : constant := 60.0;
  x_start_limit      : constant := right_border_limit - left_border_limit;
  target_start_y     : constant := 3960.0;
  target_start_z     : constant := 0.0;
end Parameter_Data_Base;
```

```
-----------------------------------------------------------------------
--% UNIT:      RDL Package Body Subunit.                              --
--% Effects:   Supports all Rocket Data Link functions of Simulator.  --
--% Modifies:  No global data is modified.                            --
--% Requires:  No initialization is required.                         --
--% Raises:    No explicitly raised exceptions are propagated.        --
--% Engineer:  T. Griest.                                             --
-----------------------------------------------------------------------

--|
--| PACKAGE BODY : Simulate.RDL (Rocket Data Link)
--|
--|    The RDL package provides tasks to interface to the Rocket Data Link
--|  issuing messages for new rocket positions and receiving messages
--|  commanding new rocket attitudes.
--|

-- Modifications Log
--
-- 88-10-30 : TEG => Original created.
--

separate(Simulate)
package bodv RDL is                        -- Rocket Data Link Simulator

  stack_size : constant := 348;

  task type Rock_Sup_Type is
    pragma PRIORITY(Config.rock_sup_priority);
  end Rock_Sup_Type;
  for Rock_Sup_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                               stack_size);
  Rock_Sup : Rock_Sup_Type;

  task body Rock_Sup_Type is separate;

  task body Report_Buf_Type is separate;

  task body Guide_Buf_Type is separate;

end RDL;
```

```
-------------------------------------------------------------------
--% UNIT:     Report_Buf Task Body Subunit.                      --
--% Effects:  Buffers Rocket report data between simulator and Control. --
--% Modifies: No global data is modified.                        --
--% Requires: No initialization is required.                     --
--% Raises:   No explicitly raised exceptions are propagated.    --
--% Engineer: T. Griest.                                         --
-------------------------------------------------------------------

--|
--| TASK BODY : Simulate.RDL.Report_Buf
--|
--|   The Report_Buf task acts as a buffer between the rocket data link
--| support task Rock_Sup and the Rocket.Control task which processes
--| the rocket data.  The task contains only accept statements for
--| rendezvous purposes to allow for schedule slippage from both sides.
--|   The data is a list of the rockets new positions as they fly across
--| the battlefield.  Rocket.Control is the receiver of this list and
--| Rock_sup is the supplier.  This routine should be contrasted to Guide_buf.
--| Note that if for some reason the Rock_Sup fails to deliver new rocket
--| positions, Rocket.Control will still display the old positions (but only
--| for one interval, after which if they are still missing are considered
--| to have destroyed themselves).  Note also that even if there are no rockets
--| active, that a list is still passed with a length of zero.
--|

-- Modifications Log
--
-- 88-11-30 : TEG => Original created.
--

with Debug_IO;
with Time_Stamp;

separate (Simulate.RDL)

task body Report_Buf_Type is
  use Types;
  MSG_COUNT      : Types.WORD := 1;
  ROCKET_MSG     : Rocket.ROCKET_MSG_TYPE;
begin
  ROCKET_MSG.NUM_ROCKETS := 0;              -- default
  loop
    select
      accept Put_Report(DATA : in Rocket.ROCKET_MSG_TYPE) do
        Time_Stamp.Log(0066);     --$TP(0066) Report_Buf accept Put_Report start
        ROCKET_MSG.NUM_ROCKETS := DATA.NUM_ROCKETS;          -- copy data
        ROCKET_MSG.ROCKET_LIST(Types.WORD_INDEX(1)..DATA.NUM_ROCKETS) :=
                    DATA.ROCKET_LIST(Types.WORD_INDEX(1)..DATA.NUM_ROCKETS);
        MSG_COUNT := 1;
        Time_Stamp.Log(0067);     --$TP(0067) Report_Buf accept Put_Report end
```

```
      end Put_Report;
    or
      when MSG_COUNT = 1 =>
      accept Get_Report(DATA : out Rocket.ROCKET_MSG_TYPE) do
        Time_Stamp.Log(0068);      --$TP(0068) Report_Buf accept Get_Report start
        DATA.NUM_ROCKETS := ROCKET_MSG.NUM_ROCKETS;
        DATA.ROCKET_LIST(Types.WORD_INDEX(1)..ROCKET_MSG.NUM_ROCKETS) :=
            ROCKET_MSG.ROCKET_LIST(Types.WORD_INDEX(1)..ROCKET_MSG.NUM_ROCKETS);
        MSG_COUNT := 0;
        Time_Stamp.Log(0069);      --$TP(0069) Report_Buf accept Get_Report end
      end Get_Report;
    end select;
  end loop;
exception
  when others =>
    Debug_IO.Put_Line("REPORT_BUF termination due to exception.");
end Report_Buf_Type;
```

```
-----------------------------------------------------------------
--% UNIT:     Rocket Package Spec.                              --
--% Effects:  Provides structure for Rocket managment within BDS. --
--% Modifies: No global data is modified.                       --
--% Requires: No initialization is required.                    --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: T. Griest.                                        --
-----------------------------------------------------------------

--|
--| PACKAGE SPEC : Rocket
--|
--|   This package contains the declaration to the control task which is
--| the main rocket processing task.  It also declares the two main types
--| used for processing rocket information.
--|

-- Modifications Log
--
-- 88-11-05 : TEG => Original created.
--

with Types;
with Config;
with Sync;

package Rocket is

stack_size             : constant := 1936;      -- in bytes

-------------------------------------------
--         REPORT INFORMATION            --
-------------------------------------------
  type ROCKET_ITEM_TYPE is record       -- provides essentials on a rocket
      SYNC_TAG          : Sync.SEQ_TYPE;
      ROCKET_ID         : Types.WORD_INDEX;
      POSITION          : Types.POSITION_TYPE;
    end record;

  type ROCKET_LIST_TYPE is        -- list of all rocket data
    array(Types.WORD_INDEX range <>) of ROCKET_ITEM_TYPE;

  type ROCKET_MSG_TYPE is record
      NUM_ROCKETS       : Types.WORD_INDEX;
      ROCKET_LIST       : ROCKET_LIST_TYPE(1..Config.max_rockets);
  end record;

-------------------------------------------
--         GUIDANCE INFORMATION          --
-------------------------------------------
  type ROCKET_GUIDE_TYPE is record
```

```
    ROCKET_ID              : Types.WORD_INDEX;
    AIMPOINT               : Types.AIMPOINT_TYPE;
  end record;

  type ROCKET_GUIDE_LIST_TYPE is      -- list of all guidance data
    array(Types.WORD_INDEX range <>) of ROCKET_GUIDE_TYPE;

  type ROCKET_GUIDE_MSG_TYPE is record
    NUM_ROCKETS            : Types.WORD_INDEX;
    ROCKET_GUIDE_LIST      : ROCKET_GUIDE_LIST_TYPE(1..Config.max_rockets);
  end record;

  task type Control_Type is                   -- for overall engagement control
    entry Start;
    entry Get_Next_Report(ROCKET_REPORT_MSG : in ROCKET_MSG_TYPE);
    pragma PRIORITY(Config.control_priority);
  end Control_Type;
  for Control_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                                      stack_size);
  Control                  : Control_Type;


end Rocket;   -- package specification
```

```
-----------------------------------------------------------------
--% UNIT:     Rocket Package Body.                                --
--% Effects:  Provides structure for Rocket managment within BDS.  --
--% Modifies: No global data is modified.                         --
--% Requires: No initialization is required.                      --
--% Raises:   No explicitly raised exceptions are propagated.     --
--% Engineer: T. Griest.                                          --
-----------------------------------------------------------------

--|
--| PACKAGE BODY : Rocket
--|
--|   The Rocket package provides all processing to maintain the rockets
--| in flight.
--| The rocket guidance activity is given overall control by the Control task.
--| "Control" is used to accept rocket reports, and is responsible for engaging
--| the targets, providing updates to the Graphics.Display task, and generating
--| the guidance messages for the Rocket Data Link.  It achieves much of this
--| with the assistance of one (or more) Guidance task(s).  The Guidance task
--| is responsible for taking a set of the rockets and producing a new
--| aimpoint for each rocket/target in that set.  The activities of the
--| guidance task(s), as well as the Control task can be overlapped
--| considerably, and therefore may benefit from the addition of processors.
--|

-- Modifications Log
--
-- 88-11-25 : TEG => Original created.
-- 89-11-22 : MPS => Aimpoint_Info type created to allow less traffic on the
--                   net.
--
with Debug_IO;
with Status;      -- maintains number Rockets Active
with Shapes;                                  -- for rocket shapes
with Graphics;    -- for graphics operations/colors
with Distrib;

package body Rocket is

guidance_stack_size            : constant := 660;

  GUIDANCE_LIST_ERROR  : exception; -- if guidance list does not match history
--
--  This history data is provided to a guidance task, which in turn processes
--  it and returns the next guidance information needed for each rocket.
--
  type POSITION_DATA_TYPE is record   -- containing rocket/target information
    ACTIVE               : BOOLEAN;        -- if rocket was previously active
    ROCKET_POS           : Types.POSITION_TYPE; -- latest rocket position
    TARGET_POS           : Types.POSITION_TYPE; -- latest target position
  end record;
```

```
  type POSITION_LIST_TYPE is
                      array(Types.WORD_INDEX range <>) of POSITION_DATA_TYPE;

  type AIMPOINT_LIST_TYPE is
                      array(Types.WORD_INDEX range <>) of Types.AIMPOINT_TYPE;

  AIMPOINT_INFO    : POSITION_LIST_TYPE(1..Config.max_rockets);

  NEXT_GUIDE_MSG   : ROCKET_GUIDE_MSG_TYPE;

  task type Guidance_Type is
    entry History(AIM_DATA : in POSITION_LIST_TYPE);
    entry Next_Guidance(AIMPOINT_LIST : out AIMPOINT_LIST_TYPE);
    pragma PRIORITY(Config.guidance_priority);
  end Guidance_Type;
  for Guidance_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                                guidance_stack_size);

  Rocket_Guide : array(Types.WORD_INDEX range 1..Distrib.num_guide_tasks)
                                                      of Guidance_Type;

  task body Guidance_Type is separate;

  task body Control_Type is separate;

end Rocket;  -- package body
```

```
-----------------------------------------------------------------
--% UNIT:      Rock_Sup Task Body Subunit.                      --
--% Effects:  Provides all Rocket Support for Simulator, including     --
--%            target intercept detection.                      --
--% Modifies: Updates state of rockets and targets in Simulator DBase. --
--% Requires: No initialization is required.                    --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: T. Griest.                                        --
-----------------------------------------------------------------

--
-- Copyright(C) 1988, LabTek Corporation.  Permission is granted to copy
-- and/or use this software provided that this copyright notice is included
-- and all liability for its use is accepted by the user.
--

--|
--| TASK BODY : Simulate.RDL.Rock_Sup
--|
--|   The rocket support task provides the necessary rocket motion, based
--| on previous position and the application of a new guidance aimpoint.
--| It generates a new report "ROCKET_MSG" for a buffer task (Report_Buf)
--| to forward to the BDS Rocket.Control task.  Likewise, the Rocket.Control
--| task issues guidance messages to the buffer task (Guide_Buf) which are
--| made available to the Rock_Sup task.  ROCKET/TARGET intercepts are
--| checked in the shared data base within the simulator.  In such cases,
--| both the rocket and target are destroyed (marked inactive).
--|

-- Modifications Log
--
-- 88-12-05 : TEG => Original created.
-- 89-09-12 : MPS => Changed call to Traject to reflect new flight dynamics.
--

with Traject;           -- trajectory planner
with Calendar;
with Interrupt_Control;
with Time_Stamp;
with Sync;
pragma ELABORATE(Traject, Calendar, Interrupt_Control, Time_Stamp);

separate (Simulate.RDL)

task body Rock_Sup_Type is

  use Calendar;          -- for - operator
  use Types;             -- for operators
  use Sync;              -- for sequence operations
  start_position        : constant Types.POSITION_TYPE :=
                             (Config.launch_x,Config.launch_y,Config.launch_z);
```

```
 ROCKET_MSG           : Rocket.ROCKET_MSG_TYPE;
 GUIDE_MSG            : Rocket.ROCKET_GUIDE_MSG_TYPE;
 GUIDE_MSG_INDEX      : Types.WORD_INDEX;
 REPORT_MSG_INDEX     : Types.WORD_INDEX;
 POSITION             : Types.POSITION_TYPE;  -- temp
 SEQUENCE_TAG         : Sync.SEQ_TYPE := 0;
 START_TIME           : Calendar.TIME;
 DELAY_PERIOD         : DURATION;
--
--  MAKE_REPORT: process current rocket ID
--
 procedure Make_Report(ID : Types.WORD_INDEX; POS : Types.POSITION_TYPE) is
                            -- checks if rocket has collided with
                            -- any targets or ground.  If so, delete
                            -- target(s) and rocket.
   DELTA_X             : Types.LONG_FIXED;
   DELTA_Y             : Types.LONG_FIXED;
   DELTA_Z             : Types.LONG_FIXED;
   DELTA_T             : Types.LONG_FIXED;  -- time for rocket to reach ground
   ROCKET_POS          : Types.POSITION_TYPE;
 begin   -- of  Make_Report
   --$TP(0070) Rock_Sup.Make_Report start
   if POS.Z < 0.0 then
     ROCKETS(ID).ACTIVE := FALSE;           -- destroy rocket
--
-- compute time it took to get to zero
--
     DELTA_X := POS.X - ROCKETS(ID).POSITION.X;
     DELTA_Y := POS.Y - ROCKETS(ID).POSITION.Y;
     DELTA_Z := POS.Z - ROCKETS(ID).POSITION.Z;

     if DELTA_Z = 0.0 then
       DELTA_T := 0.0;
     else
       DELTA_T := Types.LONG_FIXED(ROCKETS(ID).POSITION.Z/abs(DELTA_Z));
     end if;
--
-- find terminal position of Rocket
--
     ROCKET_POS.X := ROCKETS(ID).POSITION.X + DELTA_T*DELTA_X;
     ROCKET_POS.Y := ROCKETS(ID).POSITION.Y + DELTA_T*DELTA_Y;
--TBD since targets are always at Z=0, collision point is always 0
--       ROCKET_POS.Z := ROCKETS(ID).POSITION.Z + Types.METERS(DELTA_T*DELTA_Z);
--
--  Now search target list to see if any targets within "kill_radius"
--  perimeter of rocket
--
     for TARGET_ID in TARGETS'range loop
       Interrupt_Control.Disable;                -- access to shared data
       if TARGETS(TARGET_ID).ACTIVE then
         DELTA_X := ROCKET_POS.X - TARGETS(TARGET_ID).POSITION.X;
```

```
        DELTA_Y := ROCKET_POS.Y - TARGETS(TARGET_ID).POSITION.Y;
--TBD should use distance DISTANCE := Math.Sqrt( Types.METERS(DELTA_X*DELTA_X) +
--TBD                                Types.METERS(DELTA_Y*DELTA_Y) +
--TBD                                Types.METERS(DELTA_Z*DELTA_Z)));
        if abs DELTA_X < Config.kill_radius and  -- this makes square box
           abs DELTA_Y < Config.kill_radius      -- around each target
        then
          TARGETS(TARGET_ID).ACTIVE := FALSE;  -- destroy target
        end if;
      end if;
      Interrupt_Control.Enable;
    end loop;
  else                        -- Rocket did not hit ground or target
    REPORT_MSG_INDEX := REPORT_MSG_INDEX + 1;
    ROCKET_MSG.ROCKET_LIST(REPORT_MSG_INDEX) := (SEQUENCE_TAG,ID,POS);
  end if;
  --$TP(0071) Rock_Sup.Make_Report end
 end Make_Report;

---------------------------------------------------
--        ROCKET SUPPORT TASK BODY               --
---------------------------------------------------
begin
 for ID in ROCKETS'range loop                -- initialize to all inactive
   ROCKETS(ID).ACTIVE := FALSE;
 end loop;
 START_TIME := Calendar.CLOCK;               -- find out when xeq begins

 loop
   --$TP(0072) Rock_Sup task start
   START_TIME := START_TIME + Config.interval;
   if SEQUENCE_TAG = Sync.SEQ_TYPE'last then -- update TIME_TAG to be able
     SEQUENCE_TAG := 0;                      --  to differentiate between
   else                                      --  stale and new reports
     SEQUENCE_TAG := SEQUENCE_TAG + 1;
   end if;

   --$TP(0073) Rock_Sup rendezvous with Guide_Buf start
   RDL.Guide_Buf.Get_Guide(GUIDE_MSG);       -- fetch latest guidance message
   --$TP(0074) Rock_Sup rendezvous with Guide_Buf end
--
--  Go through each rocket, and if active, apply trajectory to
--  current position for 1 interval.
--
   GUIDE_MSG_INDEX := 1;                   -- pointer msg.rocket_guide_list
   REPORT_MSG_INDEX := 0;
   for ROCKET_ID in ROCKETS'range loop
     if GUIDE_MSG_INDEX <= GUIDE_MSG.NUM_ROCKETS and then
        ROCKET_ID = GUIDE_MSG.ROCKET_GUIDE_LIST(GUIDE_MSG_INDEX).ROCKET_ID
     then
--
```

```
--  This rocket is in the list, see if it was previously active
--
      if not ROCKETS(ROCKET_ID).ACTIVE then
--
-- filter out guidance messages for rockets that have recently been
-- destroyed (but BDS doesn't know it yet)
--
        if GUIDE_MSG.ROCKET_GUIDE_LIST(GUIDE_MSG_INDEX).AIMPOINT.ELEVATION =
           Config.launch_elevation
        then        -- a new launch
          ROCKETS(ROCKET_ID).ACTIVE := TRUE; -- launch
          ROCKETS(ROCKET_ID).POSITION := start_position;
          Make_Report(ROCKET_ID,start_position);  -- start at launcher
        end if;
      else
--
--  Now compute new X,Y,Z position.
--
        Traject.Get_New_Position(ROCKET_ID,
                      GUIDE_MSG.ROCKET_GUIDE_LIST(GUIDE_MSG_INDEX).AIMPOINT,
                      POSITION);
        Make_Report(ROCKET_ID,POSITION);
        ROCKETS(ROCKET_ID).POSITION := POSITION;
      end if;   -- rocket active check
      GUIDE_MSG_INDEX := GUIDE_MSG_INDEX + 1;
    else                                    -- no guidance for this rocket
      if ROCKETS(ROCKET_ID).ACTIVE then
--
-- no guidance information for active rocket, simply don't move it
--
        POSITION := ROCKETS(ROCKET_ID).POSITION;
        Make_Report(ROCKET_ID,POSITION);
      end if;      -- rocket active check
    end if;        -- guide entry exists check
  end loop;
--
-- New report list has been generated.  Send it to buffer task.
--
  ROCKET_MSG.NUM_ROCKETS := REPORT_MSG_INDEX;
  --$TP(0075) Rock_Sup rendezvous with Report_Buf start
  RDL.Report_Buf.Put_Report(ROCKET_MSG);    -- issue next rocket report
  --$TP(0076) Rock_Sup rendezvous with Report_Buf end
--
-- Delay to make rocket motion reports periodic
--
  DELAY_PERIOD := START_TIME - Calendar.CLOCK;
  if DELAY_PERIOD < 0.0 then
    START_TIME := CLOCK;
  end if;
  --$TP(0077) Rock_Sup task end
  delay DELAY_PERIOD;
```

```
  end loop;
end Rock_Sup_Type;
```

```
-------------------------------------------------------------------
--% UNIT:     Sensor Package Body Subunit.                        --
--% Effects:  Provides structure for all simulator Target motion.  --
--% Modifies: Simulator target data is updated.                    --
--% Requires: Initialization is performed by Sensor.Initialize.    --
--% Raises:   TARGET_CREATE_ERROR is raised if no room for more targets.--
--% Engineer: M. Sperry.                                           --
-------------------------------------------------------------------

--|
--| PACKAGE BODY : Simulate.Sensor
--|
--|   The sensor package supports the targ_sup task by keeping a history
--| of the old target position, the current X velocity of the target, a
--| desired X velocity of the target, how long to stay at that desired velocity
--| and finally the attributes of the class of the target.  The Y velocity
--| is constant with respect to the class of the target, as is the turning
--| frequency of the X direction.
--|

-- Modifications Log
--
-- 88-10-22 : TEG => Original created.
-- 89-11-27 : MPS => Get_New_ID, Get_New_Position, and Activate_Target created.
--

with Interrupt_Control;
with Math;
with Parameter_Data_Base;
with Time_Stamp;
with Debug_IO;
  pragma ELABORATE(Math, Debug_IO, Interrupt_Control);

separate(Simulate)

package body Sensor is                    -- Target Sensor Simulator
use Types;

type HISTORY_REC is record
  OLD_POS          : Types.POSITION_TYPE;
  CURRENT_VEL_X    : Types.METERS;
  DESIRED_VEL_X    : Types.METERS; -- generated randomly every CHANGE_DIR_FREQ
  CHANGED_VEL_TIME : Types.WORD_INDEX; -- intervals since DESIRED was changed
  ATTRIBUTES       : Parameter_Data_Base.TARGET_PARAMETER_TYPE;
end record;

type HISTORY_TYPE is array(Types.TARGET_INDEX_TYPE) of HISTORY_REC;

TARGET_HISTORY      : HISTORY_TYPE;
LAST_USED_TARGET_ID : Types.TARGET_INDEX_TYPE;
```

```
procedure Initialize is

--|
--| SUBPROGRAM BODY : Simulate.Sensor.Initialize
--|
--|  Initialize is responsible for setting the LAST_USED_TARGET_ID to the first
--| allowable value of that type.  Also, it sets all targets to an inactive
--| (FALSE) state.
--|

begin
  LAST_USED_TARGET_ID := Types.TARGET_INDEX_TYPE'first;
  for ID in Types.TARGET_INDEX_TYPE loop
    TARGETS(ID).ACTIVE := FALSE;
  end loop;
end Initialize;

function Get_New_ID return Types.TARGET_INDEX_TYPE is

--|
--| SUBPROGRAM BODY : Simulate.Sensor.Get_New_ID
--|
--|  To simplify the code in Targ_Sup, this function keeps track of the last
--| target id used (a package level variable in Sensor.ab) and returns a new
--| target id that is not currently being used.  The target id's rollover at
--| Config.max_targets.
--|

TARGET_ID           : Types.TARGET_INDEX_TYPE;
TARGET_CREATE_ERROR : EXCEPTION;

begin
  TARGET_ID := LAST_USED_TARGET_ID;
  loop      -- loop through each target_id starting from LAST_USED_TARGET_ID
    if not TARGETS(TARGET_ID).ACTIVE then
      LAST_USED_TARGET_ID := TARGET_ID;
      if LAST_USED_TARGET_ID = Types.TARGET_INDEX_TYPE'last then
        LAST_USED_TARGET_ID := Types.TARGET_INDEX_TYPE'first;
      end if;
      exit;
    else
      if TARGET_ID = Config.max_targets then
        TARGET_ID := Types.TARGET_INDEX_TYPE'first;
      else
        TARGET_ID := TARGET_ID + 1;
      end if;
      if TARGET_ID = LAST_USED_TARGET_ID then     -- no more room for targets,
        raise TARGET_CREATE_ERROR;                -- but told to create one
      end if;
    end if;
  end loop;
```

```
  return TARGET_ID;
exception
  when TARGET_CREATE_ERROR =>
    Debug_IO.Put("TARGET_CREATE_ERROR raised in Simulate.Sensor.Get_New_ID");
end Get_New_ID;

procedure Activate_Target(TARGET_ID : Types.TARGET_INDEX_TYPE) is

--|
--| SUBPROGRAM BODY : Simulate.Sensor.Activate_Target
--|
--|  Activate_Target initializes the record which controls the target's
--| history.  It also assigns a random new starting position (in X only,
--| the starting Y and Z positions are fixed) and a new class.  The class
--| is chosen randomly via the package Math.
--|

NUM_OF_CLASSES : Types.WORD_INDEX;
CLASS          : Types.TARGET_CLASS_TYPE;
POS_X          : Types.METERS;
MAX_X_VEL      : Types.METERS;

begin
--
-- Limit the access to the shared data base in Simulate.
--
  Interrupt_Control.Disable;
--
-- Initialize Simulate.TARGETS data base.
--
  TARGETS(TARGET_ID).ACTIVE := TRUE;
  TARGETS(TARGET_ID).POSITION.Y := Types.LONG_FIXED(
                                        Parameter_Data_Base.target_start_y);
  TARGETS(TARGET_ID).POSITION.Z := Types.LONG_FIXED(
                                        Parameter_Data_Base.target_start_z);
  POS_X := Math.Get_Random_Num(Types.METERS(
                                        Parameter_Data_Base.x_start_limit));
  POS_X := POS_X + Parameter_Data_Base.left_border_limit;
  TARGETS(TARGET_ID).POSITION.X := Types.LONG_FIXED(POS_X);
  NUM_OF_CLASSES := Types.WORD_INDEX(Types.TARGET_CLASS_TYPE'pos(
                                    Types.TARGET_CLASS_TYPE'last));
  CLASS := Types.TARGET_CLASS_TYPE'val(Math.Get_Random_Num(NUM_OF_CLASSES+1));
  TARGETS(TARGET_ID).TARGET_CLASS := CLASS;
--
-- Enable the interrupts again.
--
  Interrupt_Control.Enable;
--
-- Initialize Sensor.TARGET_HISTORY data base.
--
  TARGET_HISTORY(TARGET_ID).OLD_POS := TARGETS(TARGET_ID).POSITION;
```

```
  TARGET_HISTORY(TARGET_ID).ATTRIBUTES :=
                                   Parameter_Data_Base.TARGET_PARAMS(CLASS);
  MAX_X_VEL := TARGET_HISTORY(TARGET_ID).ATTRIBUTES.MAX_VELOCITY_X;
  TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X := Math.Get_Random_Num(
                                             Types.METERS(2 * MAX_X_VEL));
  TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X := MAX_X_VEL -
                           TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X;
  TARGET_HISTORY(TARGET_ID).CHANGED_VEL_TIME := 0;
  TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X := Math.Get_Random_Num(
                                             Types.METERS(2 * MAX_X_VEL));
  TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X := MAX_X_VEL -
                           TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X;
end Activate_Target;

procedure Get_New_Position(TARGET_ID : Types.TARGET_INDEX_TYPE) is

--|
--| SUBPROGRAM BODY : Sensor.Get_New_Position
--|
--|  Get_New_Position is responsible for updating the history of the targets
--| and more importantly to return to TARGETS a new target position in the
--| Types.POSITION_TYPE, which is made up of Types.LONG_FIXED.  The target
--| is not allowed to leave the battlefield border area, therefore it changes
--| directions before bouncing against the side of the border.
--|

DIR_FREQ       : Types.WORD_INDEX;
MAX_X_VEL      : Types.METERS;
MAX_Y_VEL      : Types.METERS;
DELTA_X        : Types.METERS;
CLASS          : Types.TARGET_CLASS_TYPE;
INTERVALS_LEFT : Types.METERS;
X_POS_EST      : Types.METERS;

begin
  Time_Stamp.Log(0122);     --$TP(0122) Sensor.Get_New_Position start
--
-- Place often used but complex address calculation type variables in
-- local space.
--
  CLASS := TARGETS(TARGET_ID).TARGET_CLASS;
  DIR_FREQ := TARGET_HISTORY(TARGET_ID).ATTRIBUTES.CHANGE_DIR_FREQ;
  MAX_X_VEL := TARGET_HISTORY(TARGET_ID).ATTRIBUTES.MAX_VELOCITY_X;
  MAX_Y_VEL := TARGET_HISTORY(TARGET_ID).ATTRIBUTES.MAX_VELOCITY_Y;
  DELTA_X := TARGET_HISTORY(TARGET_ID).ATTRIBUTES.DELTA_VELOCITY_X;
--
-- Check to see if it is time to change dir.
--
  if TARGET_HISTORY(TARGET_ID).CHANGED_VEL_TIME = DIR_FREQ then
--
-- Time to change the X direction.  The DESIRED_VEL_X is a random number of
```

```
-- Types.METERS between +MAX_X_VEL and -MAX_X_VEL.
--
    TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X := Math.Get_Random_Num(
                                           Types.METERS(2 * MAX_X_VEL));
    TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X := MAX_X_VEL -
                            TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X;
    TARGET_HISTORY(TARGET_ID).CHANGED_VEL_TIME := 0;
  end if;
--
-- Increment the counter that keeps track of when it is time to change direction
--
  TARGET_HISTORY(TARGET_ID).CHANGED_VEL_TIME :=
                            .ARGET_HISTORY(TARGET_ID).CHANGED_VEL_TIME + 1;
--
-- Avoid hitting the battlefield border area.
--
  if TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X > 0.0 then
    if TARGETS(TARGET_ID).POSITION.X < Parameter_Data_Base.left_border_limit
    and TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X > 0.0 then  -- going left
      TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X :=
                                  -TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X;
    end if;
  else
    if TARGETS(TARGET_ID).POSITION.X > Parameter_Data_Base.right_border_limit
    and TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X < 0.0 then  -- going right
      TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X :=
                                  -TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X;
    end if;
  end if;
--
-- Adjust the CURRENT_VEL_X by DELTA_X if need be.
--
  if TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X <
     TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X
  then
    TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X :=
                            TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X + DELTA_X;
  elsif TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X >
        TARGET_HISTORY(TARGET_ID).DESIRED_VEL_X
  then
    TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X :=
                            TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X - DELTA_X;
  end if;
--
-- Ascertain the new position based on the velocities of the class, saving
-- the old position first, and guarantee mutally exclusive access.
--
  TARGET_HISTORY(TARGET_ID).OLD_POS := TARGETS(TARGET_ID).POSITION;

  Interrupt_Control.Disable;
  if TARGETS(TARGET_ID).ACTIVE then
```

```
    TARGETS(TARGET_ID).POSITION.X := TARGETS(TARGET_ID).POSITION.X -
                    Types.LONG_FIXED(TARGET_HISTORY(TARGET_ID).CURRENT_VEL_X);
    TARGETS(TARGET_ID).POSITION.Y :=
                    TARGETS(TARGET_ID).POSITION.Y - Types.LONG_FIXED(MAX_Y_VEL);
--
-- The Z direction is not currently implemented for targets.
--
    TARGETS(TARGET_ID).POSITION.Z := Types.LONG_FIXED(0.0);
  end if;
  Interrupt_Control.Enable;
  Time_Stamp.Log(0123);      --$TP(0123) Sensor.Get_New_Position end
end Get_New_Position;

task body Targ_Sup_Type is separate;

end Sensor;          -- body
```

```
------------------------------------------------------------------------
--% UNIT:      Shapes Package Spec.                                    --
--% Effects:   Provides all graphics symbology.                        --
--% Modifies:  No global data is modified.                             --
--% Requires:  No initialization is required.                          --
--% Raises:    No explicitly raised exceptions are propagated.         --
--% Engineer:  T. Griest / M. Sperry.                                  --
------------------------------------------------------------------------

--|
--| PACKAGE SPEC : Shapes
--|
--|   Package Shapes is responsible for determining the relative offsets
--| which define the shapes of all the possible symbols that can be drawn.
--| The reticle is the pointing box that is controlled by the mouse.  Note
--| that where the coordinate (0,0) is defined in terms of the shapes of
--| the object.  For example the nose of the rocket is considered to be
--| explosive in our case.  Therefore the nose of the rocket has the coordinates
--| (0,0).  Likewise, the center of the target has the (0,0) coordinate.  The
--| objects are all manipulated by their absolute coordinates or rather,
--| the coordinate (0,0).
--|

-- Modifications Log
--
-- 88-10-12 : MPS => Original created.
-- 89-08-08 : MPS => Adjusted to work with new DDC compiler
--

with Types;
with Config;

package Shapes is

type SYMBOL_TYPE is (ROCKET, TARGET, RETICLE, DOT, ZERO, ONE, TWO, THREE, FOUR,
                     FIVE, SIX, SEVEN, EIGHT, NINE, HORIZONTAL, VERTICAL);

type PIXEL is record
 X:Types.COORDINATE range Config.entire_screen_left..Config.entire_screen_right;
 Y:Types.COORDINATE range Config.entire_screen_top..Config.entire_screen_bottom;
end record;

type REL_PIXEL is record                      -- offset from base of pixel
  X_OFFSET : Types.REL_COORDINATE;                -- positive goes right
  Y_OFFSET : Types.REL_COORDINATE;                -- positive goes down
end record;

type PIXEL_LIST is array(Types.WORD_INDEX range <>) of REL_PIXEL;

type OBJECT_PTR is access PIXEL_LIST;
```

```
reticle_left   : constant := -5;             -- constants used to check if
reticle_right  : constant :=  5;             -- reticle going past screen
reticle_top    : constant := -5;             -- boundaries.
reticle_bottom : constant :=  5;
--
--  The following two constants determine how far the target center can
--  be in meters from the indicated reticle center and still allow
--  aquisition of the target for launching a rocket.  They are not the
--  same in X and Y, since the reticle is slightly rectangular.
--
reticle_x_error: constant := 40.25;     -- METERS to allow target aquisition
reticle_y_error: constant := 49.50;     -- METERS to allow target aquisition

NUMERIC          : array(0..9) of SYMBOL_TYPE := (ZERO, ONE, TWO, THREE, FOUR,
                                       FIVE, SIX, SEVEN, EIGHT, NINE);

number_width   : constant := 8;              -- widest number in pixels

m1             : constant := -1;
m2             : constant := -2;
m3             : constant := -3;
m4             : constant := -4;
m5             : constant := -5;
m6             : constant := -6;
m7             : constant := -7;
m8             : constant := -8;

OBJECT_PTR_TABLE : array(SYMBOL_TYPE) of OBJECT_PTR :=
  (TARGET => new PIXEL_LIST'(
                        (0,m2),
                     (m1,m1), (1,m1),
                  (m2,0), (0,0), (2,0),
                     (m1,1), (1,1),
                        (0,2) ),

   ROCKET => new PIXEL_LIST'(
                     (0,0),
                     (0,1),
                     (0,2),
                     (0,3),
                (m1,4),   (1,4)  ),

   RETICLE => new PIXEL_LIST'(
(m5,m5),(m4,m5),(m3,m5),                           (3,m5),(4,m5),(5,m5),
(m5,m4),                                                         (5,m4),
(m5,m3),                   (0,m3),                               (5,m3),
                           (0,m2),
                           (0,m1),
         (m3,0), (m2,0), (m1,0), (0,0),  (1,0),  (2,0),  (3,0),
                            (0,1),
                            (0,2),
```

```
(m5,3),                          (0,3),                           (5,3),
(m5,4),                                                           (5,4),
(m5,5),(m4,5),(m3,5),                                 (3,5),(4,5),(5,5)),

   DOT => new PIXEL_LIST'( (0,0),(0,0) ),

   ZERO => new PIXEL_LIST'(      (1,m8),(2,m8),(3,m8),(4,m8),(5,m8),
                           (0,m7),                                 (6,m7),
                           (0,m6),                          (5,m6),(6,m6),
                           (0,m5),                   (4,m5),       (6,m5),
                           (0,m4),            (3,m4),              (6,m4),
                           (0,m3),     (2,m3),                     (6,m3),
                           (0,m2),(1,m2),                          (6,m2),
                           (0,m1),                                 (6,m1),
                                 (1,0), (2,0), (3,0), (4,0), (5,0)),

   ONE => new PIXEL_LIST'(            (4,m8),
                           (3,m7),(4,m7),
                                  (4,m6),
                                  (4,m5),
                                  (4,m4),
                                  (4,m3),
                                  (4,m2),
                                  (4,m1),
                           (3,0), (4,0), (5,0)),

   TWO => new PIXEL_LIST'(       (1,m8),(2,m8),(3,m8),(4,m8),
                           (0,m7),                          (5,m7),
                                                            (5,m6),
                                                     (4,m5),
                                              (3,m4),
                                       (2,m3),
                                (1,m2),
                           (0,m1),
                           (0,0), (1,0), (2,0), (3,0), (4,0), (5,0)),

   THREE => new PIXEL_LIST'(      (1,m8),(2,m8),(3,m8),
                           (0,m7),                   (4,m7),
                                                     (4,m6),
                                                     (4,m5),
                                       (2,m4),(3,m4),
                                                     (4,m3),
                                                     (4,m2),
                           (0,m1),                   (4,m1),
                                 (1,0), (2,0), (3,0)),

   FOUR => new PIXEL_LIST'(                          (4,m8),
                                              (3,m7),(4,m7),
                                       (2,m6),       (4,m6),
                                (1,m5),              (4,m5),
                           (0,m4),(1,m4),(2,m4),(3,m4),(4,m4),(5,m4),
```

```
                                                    (4,m3),
                                                    (4,m2),
                                                    (4,m1),
                                          (3,0), (4,0), (5,0)),

FIVE => new PIXEL_LIST'(          (1,m8),(2,m8),(3,m8),(4,m8),(5,m8),
                          (0,m7),
                          (0,m6),
                          (0,m5),
                                  (1,m4),(2,m4),(3,m4),(4,m4),
                                                              (5,m3),
                                                              (5,m2),
                                                              (5,m1),
                          (0,0),  (1,0), (2,0), (3,0), (4,0), (5,0)),

SIX => new PIXEL_LIST'(                         (3,m8),(4,m8),
                                        (2,m7),
                                  (1,m7),
                          (0,m6),
                          (0,m5), (1,m5),(2,m5),(3,m5),
                          (0,m4),                       (4,m4),
                          (0,m3),                       (4,m3),
                          (0,m2),                       (4,m2),
                          (0,m1),                       (4,m1),
                                  (1,0), (2,0), (3,0)),

SEVEN => new PIXEL_LIST'(         (1,m8),(2,m8),(3,m8),(4,m8),(5,m8),
                          (0,m7),                             (5,m7),
                                                        (4,m6),
                                                 (3,m5),
                                          (2,m4),
                                   (1,m3),
                          (0,m2),
                          (0,m1),
                          (0,0)),

EIGHT => new PIXEL_LIST'(         (1,m8),(2,m8),(3,m8),(4,m8),
                          (0,m7),                             (5,m7),
                          (0,m6),                             (5,m6),
                          (0,m5),                             (5,m5),
                                  (1,m4),(2,m4),(3,m4),(4,m4),
                          (0,m3),                             (5,m3),
                          (0,m2),                             (5,m2),
                          (0,m1),                             (5,m1),
                                  (1,0), (2,0), (3,0), (4,0)),

NINE => new PIXEL_LIST'(          (1,m8),(2,m8),(3,m8),(4,m8),
                          (0,m7),                             (5,m7),
                          (0,m6),                             (5,m6),
                          (0,m5),                             (5,m5),
                                  (1,m4),(2,m4),(3,m4),(4,m4),(5,m4),
```

```
                                                              (5,m3),
                                                    (4,m2),
                                            (3,m1),
                        (1,0), (2,0)),
 HORIZONTAL => new PIXEL_LIST'((0, 0),(1, 0),(2, 0),(3, 0),(4, 0),(5, 0),
                              (6, 0),(7, 0),(8, 0),(9, 0),(10,0),(11,0),
                              (12,0),(13,0),(14,0),(15,0),(16,0),(17,0),
                              (18,0),(19,0),(20,0),(21,0),(22,0),(23,0),
                              (24,0),(25,0),(26,0),(27,0),(28,0),(29,0),
                              (30,0),(31,0),(32,0),(33,0),(34,0),(35,0),
                              (36,0),(37,0),(38,0),(39,0),(40,0),(41,0),
                              (42,0),(43,0),(44,0),(45,0),(46,0),(47,0),
                              (48,0),(49,0),(50,0),(51,0),(52,0),(53,0),
                              (54,0),(55,0),(56,0),(57,0),(58,0),(59,0),
                              (60,0),(61,0),(62,0),(63,0),(64,0),(65,0),
                              (66,0),(67,0),(68,0),(69,0),(70,0),(71,0),
                              (72,0),(73,0),(74,0),(75,0),(76,0),(77,0),
                              (78,0)),
  VERTICAL =>  new PIXEL_LIST'((0, 0),(0, 1),(0, 2),(0, 3),(0, 4),(0, 5),
                              (0, 6),(0, 7),(0, 8),(0, 9),(0,10),(0,11),
                              (0,12),(0,13),(0,14),(0,15)));

end Shapes;
```

```
-------------------------------------------------------------------------
--% UNIT:      Simulate Package Spec.                                   --
--% Effects:  Provides shared data base for Simulator.                  --
--% Modifies: No global data is modified.                               --
--% Requires: Individual tasks are responsible for init. of global data.--
--% Raises:   No explicitly raised exceptions are propagated.           --
--% Engineer: T. Griest.                                                --
-------------------------------------------------------------------------


--|
--| PACKAGE SPEC : Simulate
--|
--|   The Simulate package is used to provide input and output to the
--| BDS system.  It provides rocket flight paths and target generation.
--|

-- Modificiations Log
--
-- 88-10-15 : TEG => Original created.
--


with Target;
with Rocket;
with Sync;
with Config;

package Simulate is                    -- Overall simulation package

  package Sensor is                    -- Target Sensor Simulator
    stack_size : constant := 114;      -- in bytes
    task type Targ_Sup_Type is
      pragma PRIORITY(Config.targ_sup_priority);
      entry Next_Target_Msg(Data : out Target.TARGET_MSG_TYPE);
    end Targ_Sup_Type;
    for Targ_Sup_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                                              stack_size);
    Targ_Sup : Targ_Sup_Type;
  end Sensor;

  package RDL is                       -- Rocket Data Link Simulator

    report_buf_stack_size : constant := 302;      -- in bytes
    guide_buf_stack_size  : constant := 744;      -- in bytes

--
-- The Report_Buf task buffers Rocket Reports from the Rock_Sup task
-- and provides them to the Rocket.Control task
--
    task type Report_Buf_Type is
      pragma PRIORITY(Config.report_buf_priority);
      entry Put_Report(DATA : in  Rocket.ROCKET_MSG_TYPE);
```

```
      entry Get_Report(DATA : out Rocket.ROCKET_MSG_TYPE);
    end Report_Buf_Type;
    for Report_Buf_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit
                                                 * report_buf_stack_size);
    Report_Buf : Report_Buf_Type;

--
-- The Guide_Buf task buffers new Guidance messages from the Rocket.Control
-- task for delivery to the Rock_Sup task.
--
    task type Guide_Buf_Type is
      pragma PRIORITY(Config.guide_buf_priority);
      entry Put_Guide(DATA : in  Rocket.ROCKET_GUIDE_MSG_TYPE);
      entry Get_Guide(DATA : out Rocket.ROCKET_GUIDE_MSG_TYPE);
    end Guide_Buf_Type;
    for Guide_Buf_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit
                                                 * guide_buf_stack_size);
    Guide_Buf : Guide_Buf_Type;

  end RDL;

end Simulate;
```

```
--------------------------------------------------------------------------
--% UNIT:      Simulate Package Body.                                   --
--% Effects:  Provides shared data base for Simulator.                 --
--% Modifies: No global data is modified.                              --
--% Requires: Individual tasks are responsible for init. of global data.--
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
--------------------------------------------------------------------------


--|
--| PACKAGE BODY : Simulate
--|
--|   This package is responsible for implementation of the "driver"
--| for the BDS.  The simulator provides random target generation (subject
--| to a limit maximum), rocket trajectories, and target motion.
--|

-- Modifications Log
--
-- 88-10-29 : TEG => Original created.
--

with Types;

--
--  Simulator package to provide testing of BDS system
--
package body Simulate is                    -- Overall simulation package

------------------------------------------------
--   TARGET DATA                              --
------------------------------------------------
  type TARGET_SIM_TYPE is record     -- provides individual target information
    ACTIVE          : BOOLEAN;
    POSITION        : Types.POSITION_TYPE;
    TARGET_CLASS    : Types.TARGET_CLASS_TYPE;
  end record;

  type TARGETS_TYPE is
        array(Types.WORD_INDEX range 1..Config.max_targets) of TARGET_SIM_TYPE;

  TARGETS : TARGETS_TYPE;

------------------------------------------------
--   ROCKET DATA                              --
------------------------------------------------
  type ROCKET_SIM_TYPE is record    -- provides individual rocket information
    ACTIVE          : BOOLEAN;
    POSITION        : Types.POSITION_TYPE;
  end record;
```

```
  type ROCKETS_TYPE is
       array(Types.WORD_INDEX range 1..Config.max_rockets) of ROCKET_SIM_TYPE;

  ROCKETS : ROCKETS_TYPE;

  package body Sensor is separate;          -- Target Sensor Simulator

  package body RDL is separate;             -- Rocket Data Link Simulator

end Simulate;              -- body
```

```
--------------------------------------------------------------------
--% UNIT:     Status Package Spec.                                 --
--% Effects:  Maintains indicators and statistics on graphics display.  --
--% Modifies: Flags are cleared in spec. when values are displayed.     --
--% Requires: Initialization must be signaled by main for first display.--
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: M. Sperry.                                               --
--------------------------------------------------------------------

--|
--| PACKAGE SPEC : Status
--|
--|   The purpose of the Status specification package is to provide visibility
--| to the data base which holds the requests from the mouse, et. al.  The
--| requests are entered into a data table (called STATUS_CONTROL) and then
--| the table is checked to see if any updating of the statistics needs to be
--| done.  The checking of the table is done at an atomic level to prevent
--| the shared data from being corrupted at critical times. The commands are
--| processed from the mouse interrupt as mode first, then reset if there are
--| two commands to perform.
--|

-- Modifications Log
--
-- 88-11-08 : MPS => Original created.
--

with Types;
with Config;

package Status is

stack_size       : constant := 252;

type MODE_TYPE is (AUTOMATIC,MANUAL);

type STATUS_TYPE is (AIRBORNE, TRACKED, EXPENDED, DESTROYED);

subtype RESET_STATUS_TYPE is STATUS_TYPE range EXPENDED..DESTROYED;

type STATUS_RECORD is record
  DATA      : Types.WORD := 0;                  -- new statistic
  DISPLAYED : BOOLEAN := FALSE;                 -- need to display
end record;


type STATUS_TYPE_ARRAY is array(STATUS_TYPE'FIRST .. STATUS_TYPE'LAST) of
                                                         STATUS_RECORD;
--
-- define shared variables
--
```

```
MODE              : MODE_TYPE := MANUAL;
MODE_DISPLAYED    : BOOLEAN := FALSE;
STATUS_CONTROL    : STATUS_TYPE_ARRAY;
REQ_COUNT         : Types.WORD := 0;

STATUS_ERROR      : EXCEPTION;                  -- if data negative

--
-- define subprograms and tasks
--

procedure Initialize;                           -- initialization of screen

task type Update_Type is
  entry Signal;
  pragma PRIORITY(Config.update_priority);
end Update_Type;
for Update_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                          stack_size);
Update            : Update_Type;

end Status;
```

```
--------------------------------------------------------------------
--% UNIT:      Status Package Body.                                --
--% Effects:  Maintains indicators and statistics on graphics display.  --
--% Modifies: Flags are cleared in spec. when values are displayed.   --
--% Requires: Initialization must be signaled for first display.      --
--% Raises:   No explicitly raised exceptions are propagated.         --
--% Engineer: M. Sperry.                                              --
--------------------------------------------------------------------

--|
--| PACKAGE BODY : Status
--|
--|   The purpose of the status package body is the implementation of the status
--| update task.  Although operating at a low priority, the update task updates
--| the various statistics by a rendezvous with the graphics task.
--|

-- Modifications Log
--
-- 88-11-08 : MPS => Original created.
--

with Graphics;
with Interrupt_Control;
with Shapes;
with Machine_Dependent;
with Interrupt_Control;
with Debug_IO;
with Time_Stamp;
pragma ELABORATE(Graphics, Interrupt_Control, Debug_IO, Time_Stamp);

package body Status is

use Types;                                  -- for visibility to '+';

procedure Initialize is

--|
--| SUBPROGRAM BODY : Status.Initialize
--|
--|   This procedure is responsible for performing a rendezvous with graphics
--| for the purpose of printing the statistics titles.  After this has been
--| done,  this procedure signals the Status.Update task causing the initial
--| values of all the statistics to appear as well.
--|

type TITLE_REC_TYPE is record
  X,Y   : Types.WORD;
  TEXT  : STRING(1..Config.stats_title_max_length);
  COLOR : Graphics.COLOR_TYPE;
end record;
```

```
EMPTY  : constant STRING := "";
TITLES : array(1..Config.number_of_titles) of TITLE_REC_TYPE :=
        ((0,0, "Airborne   ",Graphics.status_color),
         (0,1, "  Rockets: ",Graphics.status_color),
         (0,3, "Tracked    ",Graphics.status_color),
         (0,4, "  Targets: ",Graphics.status_color),
         (0,8, "Totals     ",Graphics.status_color),
         (0,10,"Expended   ",Graphics.status_color),
         (0,11,"  Rockets: ",Graphics.status_color),
         (0,13,"Destroyed  ",Graphics.status_color),
         (0,14,"  Targets: ",Graphics.status_color),
         (0,18,"Mode:      ",Graphics.status_color),
         (0,20,"  Manual   ",Graphics.status_color),
         (0,22,"  Automatic",Graphics.status_color));

begin
  for I in 1..Config.number_of_titles loop
    Graphics.Display.Print_Titles(TITLES(I).X,TITLES(I).Y,
                                  TITLES(I).TEXT,
                                  TITLES(I).COLOR);
  end loop;
  Graphics.Display.Print_Titles(0,0,EMPTY,Graphics.status_color);
  Interrupt_Control.Disable;                 -- go atomic
  Status.REQ_COUNT := Status.REQ_COUNT + 1;  -- signal a request (print zeroes)
  Interrupt_Control.Enable;
  Status.Update.Signal;                      -- display statistics values
end Initialize;

task body Update_Type is

--|
--| TASK BODY : Status.Update
--|
--|   This task is used as a low priority task which ensures that updates to
--| the statistics are performed.  Only those stats which have changed since
--| the last update are written to the screen.
--|

use Types;                                    -- for visibility to "+"

x_start          : constant := 11;            -- column that status_box starts in x
x_end            : constant := 90;            -- end column of status_box
y_top_start_A    : constant := 307;           -- status_box top AUTOMATIC
y_bottom_start_A : constant := 322;           -- status_box bottom AUTOMATIC
y_top_start_M    : constant := 278;           -- status_box top MANUAL
y_bottom_start_M : constant := 293;           -- status_box bottom MANUAL
manual_offset    : constant := 29;            -- offset to draw status_box
box_start        : constant := 1;             -- range of components that
box_end          : constant := 4;             -- make up status_box.
base_x           : constant Types.COORDINATE := 120;-- x end of all statistics
```

```
airborne_y        : constant Types.COORDINATE := 25;  -- y location of stat
tracked_y         : constant Types.COORDINATE := 67;  -- y location of stat
expended_y        : constant Types.COORDINATE := 165; -- y location of stat
destroyed_y       : constant Types.COORDINATE := 207; -- y location of stat

y_statistics      : constant array(STATUS_TYPE'first .. STATUS_TYPE'last) of
      Types.COORDINATE := (airborne_y, tracked_y, expended_y, destroyed_y);

type STATUS_OLD is array(STATUS_TYPE'first .. STATUS_TYPE'last,
                    1 .. Config.statistics_length) of Graphics.MOVE_RECORD;

NEXT_MODE         : MODE_TYPE;
DISPLAY_REQUIRED  : BOOLEAN;
NEXT_DATA         : Types.WORD;
BOX_LIST          : Graphics.MOVE_LIST_TYPE(Types.WORD_INDEX range box_start..box_end);
DATA_OLD          : STATUS_OLD;
WORK_LIST         : Graphics.MOVE_LIST_TYPE(1 .. Config.statistics_length);
MOVE_PRIORITY     : Graphics.PRIORITY_TYPE := Graphics.LOW;


procedure Initialize is

--|
--| SUBPROGRAM BODY : Status.Update.Initialize
--|
--|   A procedure which intializes the DATA_OLD data base.  This procedure does
--| NOT cause the digits to be drawn.  Then, it initializes the status_box
--| around 'manual'.  Again, it does not cause the status_box to be drawn.  A
--| wakeup call from the main task will cause it to be drawn.
--|

begin
  for I in STATUS_TYPE'first .. STATUS_TYPE'last loop
    for J in 1 .. Config.statistics_length loop
      DATA_OLD(I,J).XY_OLD := (Types.COORDINATE(base_x),Types.COORDINATE(y_statistics(I)));
      DATA_OLD(I,J).XY_NEW := (Types.COORDINATE(base_x),Types.COORDINATE(y_statistics(I)));
      DATA_OLD(I,J).OBJECT := Shapes.ZERO;
      DATA_OLD(I,J).COLOR  := Graphics.status_color;
    end loop;
  end loop;
--
-- Now initialize top of status_box
--
  BOX_LIST(1).XY_OLD :=
            (Types.COORDINATE(x_start),Types.COORDINATE(y_top_start_A));
  BOX_LIST(1).XY_NEW :=
            (Types.COORDINATE(x_start),Types.COORDINATE(y_top_start_A));
  BOX_LIST(1).OBJECT := Shapes.HORIZONTAL;
  BOX_LIST(1).COLOR  := Graphics.status_box_color;
--
-- define bottom of status_box
```

```
--
  BOX_LIST(2).XY_OLD :=
             (Types.COORDINATE(x_start), Types.COORDINATE(y_bottom_start_A));
  BOX_LIST(2).XY_NEW :=
             (Types.COORDINATE(x_start), Types.COORDINATE(y_bottom_start_A));
  BOX_LIST(2).OBJECT := Shapes.HORIZONTAL;
  BOX_LIST(2).COLOR  := Graphics.status_box_color;
--
-- define left side of status_box
--
  BOX_LIST(3).XY_OLD :=
             (Types.COORDINATE(x_start), Types.COORDINATE(y_top_start_A));
  BOX_LIST(3).XY_NEW :=
             (Types.COORDINATE(x_start), Types.COORDINATE(y_top_start_A));
  BOX_LIST(3).OBJECT := Shapes.VERTICAL;
  BOX_LIST(3).COLOR  := Graphics.status_box_color;
--
-- define right side of status_box
--
  BOX_LIST(4).XY_OLD :=
             (Types.COORDINATE(x_end), Types.COORDINATE(y_top_start_A));
  BOX_LIST(4).XY_NEW :=
             (Types.COORDINATE(x_end), Types.COORDINATE(y_top_start_A));
  BOX_LIST(4).OBJECT := Shapes.VERTICAL;
  BOX_LIST(4).COLOR  := Graphics.status_box_color;
exception
  when others => Debug_IO.Put_Line("Exception raised in Status.Initialize");
end Initialize;


procedure Update_Box(NEXT_MODE : MODE_TYPE) is

--|
--| SUBPROGRAM BODY : Status.Update.Update_Box
--|
--|   A procedure which updates the four objects which represent the status_box
--| surrounding one of the modes.
--|

OFFSET : Types.WORD;

begin
  Time_Stamp.Log(0078);      --$TP(0078) Status.Update_Box start
  if NEXT_MODE = AUTOMATIC then            -- draw status_box at 'automatic'
    BOX_LIST(1).XY_NEW.Y := Types.COORDINATE(y_top_start_A);
    BOX_LIST(2).XY_NEW.Y := Types.COORDINATE(y_bottom_start_A);
    BOX_LIST(3).XY_NEW.Y := Types.COORDINATE(y_top_start_A);
    BOX_LIST(4).XY_NEW.Y := Types.COORDINATE(y_top_start_A);
  else                                     -- draw status_box at 'manual'
    BOX_LIST(1).XY_NEW.Y := Types.COORDINATE(y_top_start_M );
    BOX_LIST(2).XY_NEW.Y := Types.COORDINATE(y_bottom_start_M);
```

```
    BOX_LIST(3).XY_NEW.Y := Types.COORDINATE(y_top_start_M);
    BOX_LIST(4).XY_NEW.Y := Types.COORDINATE(y_top_start_M);
  end if;
--
--  Rendezvous with Graphics to draw new status_box
--
  Time_Stamp.Log(0079);      --$TP(0079) Status.Update_Box rendezvous with Graphics start
  Graphics.Display.Move(MOVE_PRIORITY, BOX_LIST(Types.WORD_INDEX range box_start..box_end));
  Time_Stamp.Log(008         --$TP(0080) Status.Update_Box rendezvous with Graphics end
--
--  Update status_box lists
--
  for I in Types.WORD_INDEX range box_start .. box_end loop
    BOX_LIST(I).XY_OLD := BOX_LIST(I).XY_NEW;
  end loop;
  Time_Stamp.Log(0081);      --$TP(0081) Status.Update_Box end
end Update_Box;


procedure Display_Digits(NEXT_DATA : in out Types.WORD;
                         STAT      : STATUS_TYPE) is

- |
--| SUBPROGRAM BODY : Status.Update.Display_Digits
--|
--|   A procedure which takes the DATA_OLD numbers, divides by 10 to get a
--| single digit.  That digit is used as an index into Shapes.NUMERIC, which
--| holds values to draw that number for Graphics.  It updates DATA_OLD in the
--| process.
--|

DIGIT       : Types.WORD;
STAT_X_LOC : Types.COORDINATE;

begin
  Time_Stamp.Log(0082);      --$TP(0082) Status.Display_Digits start
--
-- Erase previous data
--
  for I in 1 .. Config.statistics_length loop
    DATA_OLD(STAT,I).COLOR := Graphics.background_color;
    WORK_LIST(Types.WORD_INDEX(I)) := DATA_OLD(STAT,I);
  end loop;
  Time_Stamp.Log(0083);      --$TP(0083) Status.Display_Digits rendezvous with Graphics(1) start
  Graphics.Display.Move(MOVE_PRIORITY,WORK_LIST);
  Time_Stamp.Log(0084);      - $TP(0084) Status.Display_Digits rendezvous with Graphics(1) end
--
-- Move new into old, then display
--
  STAT_X_LOC := base_x;
  for I in reverse 1 .. Config.statistics_length loop
```

```
    DIGIT := NEXT_DATA mod 10;                  -- get rightmost digit
    DATA_OLD(STAT,I).OBJECT:= Shapes.NUMERIC(INTEGER(DIGIT));
    DATA_OLD(STAT,I).COLOR := Graphics.status_color;
    DATA_OLD(STAT,I).XY_NEW.X := STAT_X_LOC;
    STAT_X_LOC := STAT_X_LOC - Shapes.number_width; -- moving left
    WORK_LIST(Types.WORD_INDEX(I)) := DATA_OLD(STAT,I);
    NEXT_DATA := NEXT_DATA / 10;                -- get next digit
  end loop;
  Time_Stamp.Log(0085);      --$TP(0085) Status.Display_Digits rendezvous with Graphics(2) start
  Graphics.Display.MOVE(MOVE_PRIORITY,WORK_LIST);
  Time_Stamp.Log(0086);      --$TP(0086) Status.Display_Digits rendezvous with Graphics(2) end
  Time_Stamp.Log(0087);      --$TP(0087) Status.Display_Digits end
exception
  when others =>
    Debug_IO.Put_Line("Exception raised in Status.Display_Digits");
end Display_Digits;


--
-- body of UPDATE task
--
Begin
  Initialize;   -- inside task body call to initialize data structures, et. al.
  loop
    Time_Stamp.Log(0114);      --$TP(0114) Status task start
    Time_Stamp.Log(0115);      --$TP(0115) Status accept Signal start
    accept Signal;
    Time_Stamp.Log(0088);      --$TP(0088) Status accept Signal end
    Interrupt_Control.Enable;
    begin                                       -- exception block
      loop
        Interrupt_Control.Disable;
        DISPLAY_REQUIRED := not MODE_DISPLAYED;
        NEXT_MODE := MODE;
        MODE_DISPLAYED := TRUE;
        Interrupt_Control.Enable;
        if DISPLAY_REQUIRED then                -- update new status_box
          Update_Box(NEXT_MODE);
        end if;
        for I in STATUS_TYPE'first .. STATUS_TYPE'last loop
          Interrupt_Control.Disable;
          DISPLAY_REQUIRED := not STATUS_CONTROL(I).DISPLAYED;
          NEXT_DATA := STATUS_CONTROL(I).DATA;
          STATUS_CONTROL(I).DISPLAYED := TRUE;
          Interrupt_Control.Enable;
          if DISPLAY_REQUIRED then
            Display_Digits(NEXT_DATA,I);
          end if;
        end loop;
        Interrupt_Control.Disable;
        REQ_COUNT := REQ_COUNT - 1;
```

```
        exit when REQ_COUNT = 0;
        Interrupt_Control.Enable;
      end loop;
    Time_Stamp.Log(0089);     --$TP(0089) Status task end
    exception
      when others => Debug_IO.Put_Line("Exception raised in Status task");
    end;
  end loop;
end Update_Type;

end Status;
```

```
-------------------------------------------------------------------------
--% UNIT:      Sync Package Spec.                                      --
--% Effects:  No current use.  Will provide greater synchronize in futr.--
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
-------------------------------------------------------------------------

--|
--| PACKAGE SPEC : Sync
--|
--|   Package Sync contains a time type for use in synchronizing message
--| reception and transferring.
--|

-- Modifications Log
--
-- 88-11-25 : TEG => Original created.
-- 89-11-22 : MPS => Created the SEQ_TYPE to keep track of messages across
--                   the net synchronized with respect to time.
--

with Types;

package Sync is
  type SEQ_TYPE is new Types.WORD_INDEX;
end Sync;
```

```
------------------------------------------------------------------
--% UNIT:      Target Package Spec.                             --
--% Effects:  Provides structure for BDS Target management.     --
--% Modifies: No global data is modified.                       --
--% Requires: No initialization is required.                    --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: T. Griest.                                        --
------------------------------------------------------------------

--|
--| PACKAGE SPEC : Target
--|
--|   Package Target provides target tracking and display management.  In
--| addition, it provides the data structures necessary to keep a list
--| of targets alive.  These data bases are accessed in a guaranteed mutually
--| exclusive way, since more than one task accesses the data structures
--| declared here.  The TARGET_DATA_TYPE uses a record representation clause
--| because the number of allowed targets is a relatively large number.  This
--| number is defined in the constant Config.max_targets.  The clause reduces
--| the number of words necessary from three to one.  Although pragma PACK
--| may have also been used to limit the amount of traffic through the
--| rendezvous, it would not have been standard (i.e., the bit ordering may
--| have been different from implementation to implementation).
--|

-- Modifications Log
--
-- 88-11-12 : TEG => Original created.
--

with Types;
with Config;

package Target is

  track_stack_size      : constant := 3928;
  track_data_stack_size : constant := 1506;

  subtype TARGET_ID_TYPE is Types.WORD_INDEX range 0..Config.max_targets;

  type TARGET_ITEM_TYPE is record     -- provides individual target information
    TARGET_ID         : TARGET_ID_TYPE;
    POSITION          : Types.POSITION_TYPE;
    TARGET_CLASS      : Types.TARGET_CLASS_TYPE;
  end record;

  type TARGET_LIST_TYPE is            -- list of all available targets items
            array(Types.WORD_INDEX range <>) of TARGET_ITEM_TYPE;

  type TARGET_MSG_TYPE is record      -- incoming message from Sensor
    NUM_TARGETS        : Types.WORD_INDEX;
```

```
  TARGET_LIST         : TARGET_LIST_TYPE(Types.TARGET_INDEX_TYPE);
end record;

type TARGET_STATUS_TYPE is record
  ACTIVE              : BOOLEAN;
  ENGAGED             : BOOLEAN;
  CLASS               : Types.TARGET_CLASS_TYPE;
end record;
  for TARGET_STATUS_TYPE use record
    ACTIVE  at 0 range 0..0;
    ENGAGED at 0 range 1..1;
    CLASS   at 0 range 2..3;
  end record;

type TARGET_DATA_TYPE is record
  STATUS              : TARGET_STATUS_TYPE;
  POSITION_NEW        : Types.POSITION_TYPE;
  POSITION_OLD        : Types.POSITION_TYPE;
end record;

type TARGET_DATA_LIST_TYPE is  -- used to communicate with Rocket.Control
  array(Types.TARGET_INDEX_TYPE) of TARGET_DATA_TYPE;

task type Track_Type is
  entry Start;
  pragma PRIORITY(Config.track_priority);
end Track_Type;
for Track_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                         track_stack_size);
Track                 : Track_Type;

task type Track_Data_Type is
  entry Put(DATA           : in  TARGET_DATA_LIST_TYPE; -- put new list
            NEXT_ENGAGE    : out TARGET_ID_TYPE;        -- get new engagement
            NEXT_DISENGAGE : out TARGET_ID_TYPE); -- and disengagement

  entry Get(DATA           : out TARGET_DATA_LIST_TYPE; -- get new list
            NEXT_ENGAGE    : in  TARGET_ID_TYPE;        -- put new engagement
            NEXT_DISENGAGE : in  TARGET_ID_TYPE);       -- and disengagement
  pragma PRIORITY(Config.track_data_priority);
end Track_Data_Type;
for Track_Data_Type'STORAGE_SIZE use INTEGER(Config.bytes_per_storage_unit *
                                              track_data_stack_size);
Track_Data            : Track_Data_Type;


end Target;   -- package specification
```

```
-------------------------------------------------------------------------
--% UNIT:      Target Package Body.                                    --
--% Effects:   Provides structure for BDS Target management.           --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:    No explicitly raised exceptions are propagated.         --
--% Engineer: T. Griest.                                               --
-------------------------------------------------------------------------

--|
--| PACKAGE BODY : Target
--|
--|
--|   Package Target provides target tracking and display management.
--|

-- Modifications Log
--
-- 88-12-03 : TEG => Original created.
--


package body Target is

  task body Track_Type is separate;

  task body Track_Data_Type is separate;

end Target;   -- package body
```

```
-----------------------------------------------------------------
--% UNIT:      Targ_Sup Task Body Subunit.                      --
--% Effects:  Provides Simulator motion control for all targets.     --
--% Modifies: Modifies TARGETS and TARGET_HISTORY global data.    --
--% Requires: No initialization is required.                    --
--% Raises:   TARGET_CREATE_ERROR if told to create when max exceeded. --
--% Engineer: M. Sperry.                                         --
-----------------------------------------------------------------

--|
--| TASK BODY : Simulate.Sensor.Targ_Sup
--|
--|   A task which sends a list to the caller describing new targets and
--| targets which have made it past the bottom border of the BDS and thus are
--| considered to have been destroyed since these targets are no longer the
--| concern of the BDS.  These targets are described by not being on
--| the list.  Note that new targets are created first and then those
--| that need to be destroyed are processed.  This task is timed so that
--| the list is ready only during 100 millisecond intervals.  In an attempt
--| to generate random numbers, channel two on the timer chip is used.
--|

-- Modifications Log
--
-- 88-10-25 : MPS => Original created.
-- 89-08-08 : MPS => All references to hardware were made to point to HW_Config.
-- 89-11-29 : MPS => Re-structured Targ_Sup to use calls in body of Sensor.
--

with Calendar;
with Debug_IO;
with Time_Stamp;
with HW_Config;
with Distrib;
  pragma ELABORATE(Calendar, Debug_IO, Time_Stamp, Distrib);

separate (Simulate.Sensor)

task body Targ_Sup_Type is

use Calendar;                                  -- for visibility to "-"
use Types;                                     -- for visibility to "/" etc.

CURRENT_NUM_OF_TARGETS : Types.WORD_INDEX;        -- local count of targets
TARGET_COUNTER         : Types.WORD_INDEX;        -- Target index for array
TEMP                   : Types.POSITION_TYPE;     -- for fixed compiler bug
START_TIME             : Calendar.TIME;
DELAY_PERIOD           : DURATION;
NEW_TARGET_ID          : Types.TARGET_INDEX_TYPE;

--
```

```
-- Targ_Sup task body
--

begin
  CURRENT_NUM_OF_TARGETS := 0;   -- no targets - yet.
  Initialize;
--
-- Take the time.
--
  START_TIME := Calendar.Clock;
  loop
    Time_Stamp.Log(0092);     --$TP(0092) Targ_Sup task start
    START_TIME := START_TIME + Config.interval;
--
-- Check number of Targets; if less than maximum, then add a new
-- Target to the list.
--
    if CURRENT_NUM_OF_TARGETS < Distrib.NUM_TARGETS then
      NEW_TARGET_ID := Get_New_ID;
      Activate_Target(NEW_TARGET_ID); -- initializes TARGETS and TARGET_HISTORY
    end if;
--
-- Move each target.
--
    for ID in Types.TARGET_INDEX_TYPE loop
      if TARGETS(ID).ACTIVE then
        Get_New_Position(ID);   -- updates TARGETS(ID).POSITION
      end if;
    end loop;
--
-- See if any targets made it to the enemy line.
-- These targets are no longer the concern of the BDS. They
-- are deleted from the list.
--
    for ID in Types.TARGET_INDEX_TYPE loop
      Interrupt_Control.Disable;
      if TARGETS(ID).ACTIVE then
        if TARGETS(ID).POSITION.Y < Config.launch_y then
          CURRENT_NUM_OF_TARGETS := CURRENT_NUM_OF_TARGETS - 1;
          TARGETS(ID).ACTIVE := FALSE;
        end if;
      end if;
      Interrupt_Control.Enable;
    end loop;
--
-- Move the list into the target list kept by the target spec.
--
    Time_Stamp.Log(0093);     --$TP(0093) Targ_Sup accept Next_Target_Msg start
    accept Next_Target_Msg(DATA : out Target.TARGET_MSG_TYPE) do
      TARGET_COUNTER := 0;
      for ID in Types.TARGET_INDEX_TYPE loop
```

```
        Interrupt_Control.Disable;
        if TARGETS(ID).ACTIVE then
          TARGET_COUNTER := TARGET_COUNTER + 1;
          TEMP := TARGETS(ID).POSITION;  -- fixed compiler code bug
          DATA.TARGET_LIST(TARGET_COUNTER).POSITION := TEMP;
          DATA.TARGET_LIST(TARGET_COUNTER).TARGET_CLASS :=
                                          TARGETS(ID).TARGET_CLASS;
          DATA.TARGET_LIST(TARGET_COUNTER).TARGET_ID := ID;
        end if;
        Interrupt_Control.Enable;
      end loop;
--
-- Update number of active targets in the BDS.
--
      CURRENT_NUM_OF_TARGETS := TARGET_COUNTER;
      DATA.NUM_TARGETS := TARGET_COUNTER;
    end Next_Target_Msg;
    Time_Stamp.Log(0094);     --$TP(0094) Targ_Sup accept Next_Target_Msg end
--
-- Schedule next list out.
--
    DELAY_PERIOD := START_TIME - Calendar.Clock;
    if DELAY_PERIOD < 0.0 then
      START_TIME := Calendar.Clock;
    end if;
    Time_Stamp.Log(0095);     --$TP(0095) Targ_Sup end
    delay DELAY_PERIOD;
  end loop;
--  accept Clock(Time : in  Sync.TIME_TYPE);   --TBD
end Targ_Sup_Type;
```

```
------------------------------------------------------------------------
--% UNIT:     Track Task Body Subunit.                                  --
--% Effects:  Provides all target tracking and display for BDS.        --
--% Modifies: No global data is modified.                              --
--% Requires: No initialization is required.                           --
--% Raises:   No explicitly raised exceptions are propagated.          --
--% Engineer: T. Griest.                                               --
------------------------------------------------------------------------

--|
--| TASK BODY : Target.Track
--|
--|   The TRACK task is used to control all of the target display information.
--| It accepts data from the Sensor and maintains it for the Rocket.Control
--| task.  It is responsible for accepting the information on the targets
--| and giving that information (in the form of a Graphics.WORK_LIST) to
--| the Graphics task.  This routine can be contrasted to Rocket.Control which
--| performs many similiar functions for the rockets.
--|   Unlike the Rocket.Control task however, there is no intermediate buffer
--| task which will allow for schedule slippage like the one between the
--| Rocket.Control and the Simulate.RDL.Rock_Sup task.
--|   There is a timing loop done in this task since the rest of the system
--| derives its timing from this task and the Rocket.Control task.  It contains
--| its own timing mechanism so that if one of the tasks (or possibly another
--| processor) goes down, the entire BDS won't be locked up.
--|

-- Modifications Log
--
-- 88-10-04 : TEG => Orignal created.
--

with Graphics;
with Shapes;
with Interrupt_Control;
with Grid_to_Pixel;
with Simulate;
with Debug_IO;
with Status;
with Time_Stamp;
  pragma ELABORATE(Graphics,Shapes, Interrupt_Control,Grid_to_Pixel,
               Simulate, Debug_IO, Status, Time_Stamp);

separate (Target)
task body Track_Type is
  use Types;
  package Sensor renames Simulate.Sensor;  -- make simulation transparent
  use Types;                          -- for operators only
  TARGET_MSG      : TARGET_MSG_TYPE;
  MOVE_TARGETS    : Graphics.MOVE_LIST_TYPE(Types.TARGET_INDEX_TYPE);
  MOVE_INDEX      : Types.WORD_INDEX;
```

```
  DESTROYED         : Types.WORD;
  CREATED           : Types.WORD;
  PIXEL_POINT       : Shapes.PIXEL;
  TARGETS           : TARGET_DATA_LIST_TYPE;
  MSG_INDEX         : Types.WORD_INDEX;
  NEXT_ENGAGED      : Types.WORD_INDEX;    -- 0 if no new engagement
  NEXT_DISENGAGED : Target.TARGET_ID_TYPE;-- keep track of disengagements
  COLOR             : Graphics.COLOR_TYPE;
  ENGAGE_FLAG       : BOOLEAN;
  CLASS             : Types.TARGET_CLASS_TYPE;
  POSITION          : Types.POSITION_TYPE;  -- temp for making changes
  ESCAPED_TARGETS : Types.WORD; -- targets which made it past the BDS border
begin
  accept Start;
--
--  INITIALIZATION
--
  for I in TARGETS'range loop
    TARGETS(I).STATUS := (FALSE,FALSE,UNKNOWN);  -- init to default
  end loop;

  loop
    Time_Stamp.Log(0096);     --$TP(0096) Track task start
    Time_Stamp.Log(0097);     --$TP(0097) Track rendezvous with Targ_Sup start
    Sensor.Targ_Sup.Next_Target_Msg(TARGET_MSG);
    Time_Stamp.Log(0098);     --$TP(0098) Track rendezvous with Targ_Sup end
--
--  Zero out counters
--
    CREATED := 0;
    DESTROYED := 0;
    ESCAPED_TARGETS := 0;
--
--  Maintain history information.
--
--  Go through each target to examine its new status
--
    MSG_INDEX := 1;
    MOVE_INDEX := 0;
    for TARGET_ID in TARGETS'RANGE loop
      if TARGETS(TARGET_ID).STATUS.ACTIVE then
        if MSG_INDEX > TARGET_MSG.NUM_TARGETS or else
           TARGET_MSG.TARGET_LIST(MSG_INDEX).TARGET_ID
                                       /= TARGET_ID then -- target destroyed
--
--  Target has been destroyed, keep local accumulation of destroyed
--  targets, and add to list for Display task to erase target.
--
          DESTROYED := DESTROYED + 1;
--
--  If this target has escaped the BDS, count it in the targets which escaped.
```

```
--
        if TARGETS(TARGET_ID).POSITION_NEW.Y <= Config.launch_y then
          ESCAPED_TARGETS := ESCAPED_TARGETS + 1;
        end if;
--
-- To mark as inactive : (ACTIVE => FALSE, ENGAGED => FALSE, CLASS => UNKNOWN)
--
        TARGETS(TARGET_ID).STATUS := (FALSE, FALSE, Types.UNKNOWN);
        MOVE_INDEX := MOVE_INDEX + 1;
        PIXEL_POINT := Grid_To_Pixel(TARGETS(TARGET_ID).POSITION_NEW);
        COLOR := Graphics.background_color;
        MOVE_TARGETS(MOVE_INDEX) :=  (PIXEL_POINT,
                                      PIXEL_POINT,
                                      Shapes.TARGET,
                                      COLOR);
      else                                            -- move the target
--
--  Found a current existing target in the latest sensor report,
--  update target information and add it to move list.
--
        POSITION := TARGET_MSG.TARGET_LIST(MSG_INDEX).POSITION;
        MOVE_INDEX := MOVE_INDEX + 1;
        CLASS := TARGETS(TARGET_ID).STATUS.CLASS;
        ENGAGE_FLAG := TARGETS(TARGET_ID).STATUS.ENGAGED;
        COLOR := Graphics.target_color(CLASS, ENGAGE_FLAG);
        MOVE_TARGETS(MOVE_INDEX) :=
                 (XY_OLD => Grid_to_Pixel(TARGETS(TARGET_ID).POSITION_NEW),
                  XY_NEW => Grid_to_Pixel(POSITION),
                  OBJECT => Shapes.TARGET,
                  COLOR  => COLOR
                 );
        TARGETS(TARGET_ID).POSITION_OLD := TARGETS(TARGET_ID).POSITION_NEW;
        TARGETS(TARGET_ID).POSITION_NEW := POSITION;
        MSG_INDEX := MSG_INDEX + 1;
      end if;        -- new/old target check
    else                          -- this target wasn't previously active
      if MSG_INDEX <= TARGET_MSG.NUM_TARGETS and then
         TARGET_MSG.TARGET_LIST(MSG_INDEX).TARGET_ID
                                               = TARGET_ID then -- new target
--
--  New Target has been created, set status and put it on display
--
        CREATED := CREATED + 1;
 -- mark as active
        TARGETS(TARGET_ID).STATUS :=
                (TRUE,          -- ACTIVE
                 FALSE,         -- Engaged
                 TARGET_MSG.TARGET_LIST(MSG_INDEX).TARGET_CLASS);  -- class
        TARGETS(TARGET_ID).POSITION_OLD :=      -- set both old and new
                           TARGET_MSG.TARGET_LIST(MSG_INDEX).POSITION;
        TARGETS(TARGET_ID).POSITION_NEW :=
```

```
                          TARGET_MSG.TARGET_LIST(MSG_INDEX).POSITION;
        MOVE_INDEX := MOVE_INDEX + 1;
        CLASS := TARGETS(TARGET_ID).STATUS.CLASS;
        ENGAGE_FLAG := TARGETS(TARGET_ID).STATUS.ENGAGED;
        COLOR := Graphics.target_color(CLASS, ENGAGE_FLAG);
        MOVE_TARGETS(MOVE_INDEX) :=
                 (XY_OLD => Grid_to_Pixel(TARGETS(TARGET_ID).POSITION_OLD),
                  XY_NEW => Grid_to_Pixel(TARGETS(TARGET_ID).POSITION_NEW),
                  OBJECT => Shapes.TARGET,
                  COLOR  => COLOR
                 );
        MSG_INDEX := MSG_INDEX + 1;
      end if;                            -- end of new target check
    end if;                              -- active check
  end loop;
--
--  Now update status if any created or destroyed
--
  if CREATED /= DESTROYED or DESTROYED > 0 then
    Interrupt_Control.Disable;
    Status.STATUS_CONTROL(Status.TRACKED).DATA :=
       Status.STATUS_CONTROL(Status.TRACKED).DATA + (CREATED - DESTROYED);
    Status.STATUS_CONTROL(Status.TRACKED).DISPLAYED := FALSE;
    Status.STATUS_CONTROL(Status.DESTROYED).DATA :=
                          Status.STATUS_CONTROL(Status.DESTROYED).DATA +
                          DESTROYED - ESCAPED_TARGETS;
    Status.STATUS_CONTROL(Status.DESTROYED).DISPLAYED := FALSE;
    Status.REQ_COUNT := Status.REQ_COUNT + 1;
    if Status.REQ_COUNT = 1 then
    Time_Stamp.Log(0099);     --$TP(0099) Track rendezvous with Status start
      Status.Update.Signal;
    Time_Stamp.Log(0100);     --$TP(0100) Track rendezvous with Status end
    end if;
    Interrupt_Control.Enable;
  end if;

  Time_Stamp.Log(0101);     --$TP(0101) Track rendezvous with Track_Data start
  Target.Track_Data.Put(TARGETS,NEXT_ENGAGED,NEXT_DISENGAGED);
                                            -- send copy to Rocket.Control
  Time_Stamp.Log(0102);     --$TP(0102) Track rendezvous with Track_Data end

  if NEXT_ENGAGED > 0 then
    TARGETS(NEXT_ENGAGED).STATUS.ENGAGED := TRUE;   -- set engaged
  end if;
  if NEXT_DISENGAGED > 0 then
    TARGETS(NEXT_DISENGAGED).STATUS.ENGAGED := FALSE;
  end if;
  Time_Stamp.Log(0103);     --$TP(0103) Track rendezvous with Graphics start
  Graphics.Display.Move(Graphics.LOW, MOVE_TARGETS(1..MOVE_INDEX));
  Time_Stamp.Log(0104);     --$TP(0104) Track rendezvous with Graphics end
  Time_Stamp.Log(0105);     --$TP(0105) Track task end
```

```
  end loop;
exception
  when others =>
    Debug_IO.Put_Line("TRACK termination due to exception.");
end Track_Type;
```

```
------------------------------------------------------------------
--% UNIT:     Track_Data Task Subunit.                            --
--% Effects:  Provides buffering of target tracking data between the  --
--%           Track task and the Control task for rocket engagement.   --
--% Modifies: No global data is modified.                         --
--% Requires: No initialization is required.                      --
--% Raises:   No explicitly raised exceptions are propagated.     --
--% Engineer: T. Griest.                                          --
------------------------------------------------------------------

--|
--| TASK BODY : Target.Track_Data
--|
--|   The Track_Data task is used to buffer the most recent target list
--| from the Target.Track task and provide it to the Rocket.Control
--| task.  It also buffers new engagements or disengagements from the
--| Rocket.Control task to notify the Target.Track task that a new target
--| has been engaged or an old target destroyed.
--|   Note that only one new target can be engaged every update interval.
--| If the NEXT_ENGAGE parameter is 0, this is an invalid TARGET_ID, and
--| implies that no new target is engaged.
--|   Although there is a guard used here, it is only used for the first
--| rendezvous from Rocket.Control.  This helps the BDS system to achieve a
--| known initial state and asynchronous timing.
--|

-- Modifications Log
--
-- 88-10-11 : TEG => Original created.
--

with Time_Stamp;
with Interrupt_Control;
  pragma ELABORATE(Time_Stamp,Interrupt_Control);

separate (Target)

task body Track_Data_Type is
  use Types;

  BUFFERED_DATA       : Target.TARGET_DATA_LIST_TYPE;
  BUFFERED_ENGAGE     : Target.TARGET_ID_TYPE;
  BUFFERED_DISENGAGE  : Target.TARGET_ID_TYPE;
  DATA_COUNT          : Types.WORD := 0;
begin
--
--  Initialize local copy of data
--  initialize all target status to:
--          (ACTIVE => FALSE, ENGAGED => FALSE, CLASS => UNKNOWN)
--
  BUFFERED_ENGAGE := 0;                   -- default is no new engagement
```

```
  for I in BUFFERED_DATA'range loop
    BUFFERED_DATA(I).STATUS := (FALSE, FALSE, Types.UNKNOWN);
  end loop;
  loop
    select
      accept Put(DATA           : in  TARGET_DATA_LIST_TYPE;
                 NEXT_ENGAGE    : out TARGET_ID_TYPE;
                 NEXT_DISENGAGE : out TARGET_ID_TYPE) do
        Time_Stamp.Log(0106);      --$TP(0106) Trackdat accept Put start
        Interrupt_Control.Disable;  -- BUGFIX for RTE
        BUFFERED_DATA := DATA;
        Interrupt_Control.Enable;
        NEXT_ENGAGE := BUFFERED_ENGAGE;
        NEXT_DISENGAGE := BUFFERED_DISENGAGE;
        DATA_COUNT := 1;
        Time_Stamp.Log(0107);      --$TP(0107) Trackdat accept Put end
      end Put;
    or
      when DATA_COUNT > 0 =>
      accept Get(DATA           : out TARGET_DATA_LIST_TYPE;
                 NEXT_ENGAGE    : in  TARGET_ID_TYPE;
                 NEXT_DISENGAGE : in  TARGET_ID_TYPE) do
        Time_Stamp.Log(0108);      --$TP(0108) Trackdat accept Get start
        Interrupt_Control.Disable;  -- BUGFIX for RTE
        DATA := BUFFERED_DATA;
        Interrupt_Control.Enable;
        BUFFERED_ENGAGE := NEXT_ENGAGE;
        BUFFERED_DISENGAGE := NEXT_DISENGAGE;
        DATA_COUNT := 1;
        Time_Stamp.Log(0109);      --$TP(0109) Trackdat accept Get end
      end Get;
    end select;
  end loop;
end Track_Data_Type;
```

```
----------------------------------------------------------------------
--% UNIT:     Traject Function Spec.                                --
--% Effects:  Computes rocket motion based on previous motion and   --
--%           aimpoints received in guidance messages.              --
--% Modifies: No global data is modified.                           --
--% Requires: No initialization is required.                        --
--% Raises:   No explicitly raised exceptions are propagated.       --
--% Engineer: R. Chevier.                                           --
----------------------------------------------------------------------

--|
--| SUBPROGRAM SPEC : Traject
--|
--|   Function Traject takes the current rocket information including the
--| direction it is headed in and determines the new absolute position
--| of the rocket.  This work is done in a three dimensional system.
--|

-- Modifications Log
--
-- 88-10-29 : TEG => Original created.
-- 89-08-29 : MPS => Original replaced by R. Chevier's version.
--

with Types;

package Traject is

procedure Get_New_Position(ROCKET_ID     : Types.WORD_INDEX;
                           AIMPOINT      : Types.AIMPOINT_TYPE;
                           POS           : out Types.POSITION_TYPE);

end Traject;
```

```
------------------------------------------------------------------------
--% UNIT:      Traject Function Body.                                 --
--% Effects:   Computes rocket motion based on previous motion and    --
--%            aimpoints received in guidance messages.               --
--% Modifies:  No global data is modified.                            --
--% Requires:  No initialization is required.                         --
--% Raises:    No explicitly raised exceptions are propagated.        --
--% Engineer:  R. Chevier                                             --
------------------------------------------------------------------------

--|
--| SUBPROGRAM BODY : Traject
--|
--|   Function Traject: Is the trajectory planner for rockets and takes an
--| Azimuth, Elevation X,Y,Z position and constant velocity and returns a new
--| rocket position.
--|

-- Modifications Log
--
-- 88-12-01 : TEG => Original created.
-- 89-08-29 : MPS => Replaced original with R. Chevier's version.
-- 89-09-07 : MPS => Added the Get_New_Position function
--
with Config;
with Parameter_Data_Base;
with Rocket;
with Math;
with Time_Stamp;
  pragma ELABORATE(Math);

package body Traject is

use Types;                              -- for operators
use Math;                               -- for faster fixed math

bam_converter : constant Types.LONG_FIXED := 182.03125;

type DRIFT_RECORD_TYPE is record
  SIN_AZIMUTH   : Types.LONG_FIXED := 0.0;
  SIN_ELEVATION : Types.LONG_FIXED := 0.0;
  COS_AZIMUTH   : Types.LONG_FIXED := 0.0;
  COS_ELEVATION : Types.LONG_FIXED := 0.0;
end record;

type VELOCITY_RECORD_TYPE is record
  X : Types.LONG_FIXED := 0.0;
  Y : Types.LONG_FIXED := 0.0;
  Z : Types.LONG_FIXED := 0.0;
end record;
```

```
type LOCAL_ROCKET_REC is record
  ACTIVE     : BOOLEAN := FALSE;
  POSITION   : Types.POSITION_TYPE;
  VELOCITY   : VELOCITY_RECORD_TYPE;
  ANGLE      : Types.AIMPOINT_TYPE;
  FUEL       : Types.LONG_FIXED;
end record;

type ROCKET_HISTORY_REC is record
  LOCAL_ROCKET      : LOCAL_ROCKET_REC;
  GUIDANCE          : Rocket.ROCKET_GUIDE_TYPE;
  ROCKET_DEFAULTS   : Parameter_Data_Base.ROCKET_PARAMETER_TYPE;
  DRIFT             : DRIFT_RECORD_TYPE;
  DELTA_T           : Types.RATE_TYPE := Types.RATE_TYPE(Config.interval);
end record;

type ROCKET_HISTORY_ARRAY is array(Types.ROCKET_INDEX_TYPE) of
             ROCKET_HISTORY_REC;

ROCKET_HISTORY : ROCKET_HISTORY_ARRAY;

procedure Initialize(INDIVIDUAL_ROCKET_HISTORY : in out ROCKET_HISTORY_REC) is

begin
  INDIVIDUAL_ROCKET_HISTORY.LOCAL_ROCKET.ACTIVE := TRUE;
  INDIVIDUAL_ROCKET_HISTORY.LOCAL_ROCKET.VELOCITY := (0.0,0.0,0.0);
  INDIVIDUAL_ROCKET_HISTORY.LOCAL_ROCKET.ANGLE := (Config.launch_azimuth,
                                                   Config.launch_elevation);
  INDIVIDUAL_ROCKET_HISTORY.LOCAL_ROCKET.FUEL := Parameter_Data_Base.c_fuel;
  INDIVIDUAL_ROCKET_HISTORY.LOCAL_ROCKET.POSITION := (Config.launch_x,
                                              Config.launch_y,Config.launch_z);
end Initialize;

procedure Turn_Rocket
          (FUEL           : in out Types.LONG_FIXED;
           ROCKET_ANGLE   : in out Types.BAM;
           BDS_ANGLE      : Types.BAM;
           DELTA_T        : Types.RATE_TYPE;
           TURN_RATE      : Types.LONG_FIXED;
           TURN_BURN_RATE : Types.LONG_FIXED) is

  MAX_TURN        : Types.LONG_FIXED;
  DELTA_ANGLE     : Types.LONG_FIXED;
  FUEL_USED       : Types.LONG_FIXED;
  BURN_TIME       : Types.LONG_FIXED;
  BAMS_TURNED     : Types.LONG_FIXED;
  DEGREES_TO_TURN : Types.LONG_FIXED;

begin                                                   --Turn_Rocket
  Time_Stamp.Log(0118);      --$TP(0118) Traject.Turn_Rocket start
  DELTA_ANGLE := Types.LONG_FIXED(BDS_ANGLE) - Types.LONG_FIXED(ROCKET_ANGLE);
```

```
  if DELTA_ANGLE /= 0.0 and FUEL > 0.0 then    -- don't turn it if told not to
    MAX_TURN := DELTA_T * TURN_RATE;
--
-- If the rotation in this iteration turns the rocket too far
-- then calculate only the fuel needed to rotate the rocket
-- the required amount.
--
    DEGREES_TO_TURN := abs DELTA_ANGLE / bam_converter;
    if DEGREES_TO_TURN < MAX_TURN then
      BURN_TIME := abs DEGREES_TO_TURN / TURN_RATE;
      FUEL_USED := TURN_BURN_RATE * abs DEGREES_TO_TURN;
      ROCKET_ANGLE := BDS_ANGLE;

      --Put("BDS Angle          :"); Int_IO.Put(BDS_ANGLE); New_Line;
      --Put("Rocket Angle       :"); Int_IO.Put(ROCKET_ANGLE); New_Line;
      --Put("Delta Angle        :"); Long_Fxd_IO.Put(DELTA_ANGLE); New_Line;
      --Put("Turn Rate          :"); Long_Fxd_IO.Put(TU?N_RATE); New_Line;
      --Put("Turn Burn Rate     :"); Long_Fxd_IO.Put(TURN_BURN_RATE);New_Line;
      --Put("Burn Time          :"); Long_Fxd_IO.Put(BURN_TIME); New_Line;
      --Put("Completed Turn Fuel:"); Long_Fxd_IO.Put(FUEL_USED); New_Line;

    else
--
-- Or if the time step was not large enough for rotation
-- completion then calculate FUEL used and new ANGLE based on
-- rotation completed during this step.
--
      FUEL_USED := TURN_BURN_RATE * MAX_TURN;
      if DELTA_ANGLE < 0.0 then            -- subtract from current direction
        BAMS_TURNED := MAX_TURN * bam_converter;
        ROCKET_ANGLE := ROCKET_ANGLE - Types.BAM(BAMS_TURNED);
      else                                 -- add to current direction
        BAMS_TURNED := MAX_TURN * bam_converter;
        ROCKET_ANGLE := ROCKET_ANGLE + Types.BAM(BAMS_TURNED);
      end if;
    end if;
    FUEL := FUEL - FUEL_USED;
    if FUEL < 0.0 then
      FUEL := 0.0;
    end if;
  end if;
  Time_Stamp.Log(0119);     --$TP(0119) Traject.Turn_Rocket end
end Turn_Rocket;

procedure Calc_Trajectory
         (LOCAL_ROCKET     : in out LOCAL_ROCKET_REC;
          GUIDANCE         : Rocket.ROCKET_GUIDE_TYPE;
          ROCKET_DEFAULTS  : Parameter_Data_Base.ROCKET_PARAMETER_TYPE;
          DRIFT            : DRIFT_RECORD_TYPE;
          DELTA_T          : Types.RATE_TYPE) is
```

```
drag             : constant Types.LONG_FIXED := 0.984375;
                   -- roughly 2% of velocity per iteration
gravity          : constant Types.LONG_FIXED := 9.80665;
VX               : Types.LONG_FIXED := (LOCAL_ROCKET.VELOCITY.X);
VY               : Types.LONG_FIXED := (LOCAL_ROCKET.VELOCITY.Y);
VZ               : Types.LONG_FIXED := (LOCAL_ROCKET.VELOCITY.Z);
X                : Types.LONG_FIXED :=Types.LONG_FIXED(LOCAL_ROCKET.POSITION.X);
Y                : Types.LONG_FIXED :=Types.LONG_FIXED(LOCAL_ROCKET.POSITION.Y);
Z                : Types.LONG_FIXED :=Types.LONG_FIXED(LOCAL_ROCKET.POSITION.Z);
ELEVATION        : Types.BAM        := Types.BAM(LOCAL_ROCKET.ANGLE.ELEVATION);
AZIMUTH          : Types.BAM        := Types.BAM(LOCAL_ROCKET.ANGLE.AZIMUTH);
FUEL             : Types.LONG_FIXED := LOCAL_ROCKET.FUEL;
FORWARD_VELOCITY : Types.LONG_FIXED;
THRUST           : Types.LONG_FIXED;
TOTAL_MASS       : Types.LONG_FIXED;
DRAG_FORCE       : Types.LONG_FIXED;
AY,AX,AZ         : Types.LONG_FIXED;
SIN_ELEVATION    : Types.LONG_FIXED;
COS_ELEVATION    : Types.LONG_FIXED;
COS_AZIMUTH      : Types.LONG_FIXED;
SIN_AZIMUTH      : Types.LONG_FIXED;
TEMP_VAL         : Types.LONG_FIXED;

begin                                    -- Calc_Trajectory
  Time_Stamp.Log(0120);     --$TP(0120) Traject.Calc_Position start
  SIN_ELEVATION := Math.Sin(ELEVATION);
  SIN_AZIMUTH   := Math.Sin(AZIMUTH);
  COS_ELEVATION := Math.Cos(ELEVATION);
  COS_AZIMUTH   := Math.Cos(AZIMUTH);
  TEMP_VAL := VX*VX + VY*VY + VZ*VZ;
  FORWARD_VELOCITY := Math.Sqrt(TEMP_VAL);
--
-- Check amount of fuel left.
--
  if FUEL <= 0.0 then
    THRUST := 0.0;
  else
    THRUST := ROCKET_DEFAULTS.THRUST;
  end if;
  TOTAL_MASS := ROCKET_DEFAULTS.MASS + FUEL;
  --Put("Thrust     :"); Long_Fxd_IO.Put(THRUST); New_Line;
  --Put("Drag_Force :"); Long_Fxd_IO.PUT(DRAG_FORCE); New_Line;
  --Put("Cos_Elev   :"); Long_Fxd_IO.Put(COS_ELEVATION); New_Line;
  --Put("Sin_Az     :"); Long_Fxd_IO.Put(SIN_AZIMUTH); New_Line;
  --Put("Total_Mass :"); Long_Fxd_IO.Put(TOTAL_MASS); New_Line;
--
--   COMPUTE ACCELERATION IN EACH AXIS
--
  DRAG_FORCE := 0.0;            -- for now, null out drag acceleration
  AY := ((THRUST - DRAG_FORCE) * COS_ELEVATION) * SIN_AZIMUTH;
  AY := AY / TOTAL_MASS;
```

```
  AX := ((THRUST - DRAG_FORCE) * COS_ELEVATION) * COS_AZIMUTH;
  AX := AX / TOTAL_MASS;
  AZ := (THRUST - DRAG_FORCE) * SIN_ELEVATION;
  AZ := AZ -  TOTAL_MASS * gravity;
  AZ := AZ / TOTAL_MASS;
--
-- lose % of velocity per/iteration due to drag
--
  LOCAL_ROCKET.VELOCITY.X := DELTA_T * AX + VX * drag;
  LOCAL_ROCKET.VELOCITY.Y := DELTA_T * AY + VY * drag;
  LOCAL_ROCKET.VELOCITY.Z := DELTA_T * AZ + VZ * drag;
--
-- Update position of rocket
--
  X := X + DELTA_T * LOCAL_ROCKET.VELOCITY.X;
  LOCAL_ROCKET.POSITION.X := X;
  Y := Y + DELTA_T * LOCAL_ROCKET.VELOCITY.Y;
  LOCAL_ROCKET.POSITION.Y := Y;
  Z := Z + DELTA_T * LOCAL_ROCKET.VELOCITY.Z;
  LOCAL_ROCKET.POSITION.Z := Z;

--   New_Line;
--   Put_Line("      Velocity     Acceleration");
--   Long_Fxd_IO.Put(LOCAL_ROCKET.VELOCITY.X,6,2,0);
--   Long_Fxd_IO.Put(AX,6,2,0);
--   Put_Line("       X");
--   Long_Fxd_IO.Put(LOCAL_ROCKET.VELOCITY.Y,6,2,0);
--   Long_Fxd_IO.Put(AY,6,2,0);
--   Put_Line("       Y");
--   Long_Fxd_IO.Put(LOCAL_ROCKET.VELOCITY.Z,6,2,0);
--   Long_Fxd_IO.Put(AZ,6,2,0);
--   Put_Line("       Z");
--   New_Line;
--
-- Check for impacts to speed up code
--
  if Z > 0.0 then
--
-- When finished with the calculation update the current mass.
--
    LOCAL_ROCKET.FUEL := FUEL - DELTA_T * ROCKET_DEFAULTS.BURN_RATE;
    if LOCAL_ROCKET.FUEL < 0.0 then
      LOCAL_ROCKET.FUEL := 0.0;
    end if;
--
-- Calculate rocket turns.
--
    Turn_Rocket (LOCAL_ROCKET.FUEL,
                 LOCAL_ROCKET.ANGLE.ELEVATION,
                 GUIDANCE.AIMPOINT.ELEVATION,
                 DELTA_T, ROCKET_DEFAULTS.TURN_RATE,
```

```
                ROCKET_DEFAULTS.TURN_BURN_RATE);
    Turn_Rocket (LOCAL_ROCKET.FUEL,
                LOCAL_ROCKET.ANGLE.AZIMUTH,
                GUIDANCE.AIMPOINT.AZIMUTH,
                DELTA_T, ROCKET_DEFAULTS.TURN_RATE,
                ROCKET_DEFAULTS.TURN_BURN_RATE);
  end if;
  Time_Stamp.Log(0121);     --$TP(0121) Traject.Calc_Position end
end Calc_Trajectory;

procedure Get_New_Position(ROCKET_ID : Types.WORD_INDEX;
                           AIMPOINT  : Types.AIMPOINT_TYPE;
                           POS       : out Types.POSITION_TYPE) is

begin
  Time_Stamp.Log(0110);     --$TP(0110) Traject Start
  if not ROCKET_HISTORY(ROCKET_ID).LOCAL_ROCKET.ACTIVE then
    Initialize(ROCKET_HISTORY(ROCKET_ID));
  end if;
  ROCKET_HISTORY(ROCKET_ID).GUIDANCE.AIMPOINT := AIMPOINT;
  Calc_Trajectory(ROCKET_HISTORY(ROCKET_ID).LOCAL_ROCKET,
                  ROCKET_HISTORY(ROCKET_ID).GUIDANCE,
                  ROCKET_HISTORY(ROCKET_ID).ROCKET_DEFAULTS,
                  ROCKET_HISTORY(ROCKET_ID).DRIFT,
                  ROCKET_HISTORY(ROCKET_ID).DELTA_T);
  POS := ROCKET_HISTORY(ROCKET_ID).LOCAL_ROCKET.POSITION;
  if ROCKET_HISTORY(ROCKET_ID).LOCAL_ROCKET.POSITION.Z <= 0.0 then
    ROCKET_HISTORY(ROCKET_ID).LOCAL_ROCKET.ACTIVE := FALSE; -- kill the rocket
  end if;
  Time_Stamp.Log(0111);     --$TP(0111) Traject end
end Get_New_Position;

end Traject;
```

```
------------------------------------------------------------------
--% UNIT:     Types Package Spec.                               --
--% Effects:  Provides general purpose data types.              --
--% Modifies: No global data is modified.                       --
--% Requires: No initialization is required.                    --
--% Raises:   No explicitly raised exceptions are propagated.   --
--% Engineer: T. Griest.                                        --
------------------------------------------------------------------


--|
--| PACKAGE SPEC : Types
--|
--|   This package contains all the global types needed for the BDS and the
--| simulator.  The type WORD and its derivatives replace the type INTEGER
--| to increase portability.  The type BAM is an acronym for a Binary Angle
--| Measurement and the transformation from degrees to BAMs is performed by
--| BAMs = 32767/180 * degrees.  The BDS and the simulator use three
--| dimensional components and the screen (obviously) display of the event
--| shows it in two dimensions only.
--|

-- Modifications Log
--
-- 88-10-10 : TEG => Original created.
-- 89-08-29 : MPS => Added definitions for new rocket flight path equations.
--

with Config;

package Types is

type WORD is range -32768 .. 32767;
  for WORD'size use 16;

type WORD_INDEX is range 0 .. 32767;
  for WORD_INDEX'size use 16;

subtype ROCKET_INDEX_TYPE is WORD_INDEX range 1..Config.max_rockets;
subtype TARGET_INDEX_TYPE is WORD_INDEX range 1..Config.max_targets;

subtype COORDINATE is Types.WORD;
subtype REL_COORDINATE is Types.WORD;


type METERS is delta 0.125 range -Config.meters_in_battle_area ..
                                  Config.meters_in_battle_area;

type LONG_FIXED is delta 0.015625 range -33_554_432.0..33_554_431.0;
  for LONG_FIXED'size use 32;
--
--  RATE_TYPE is used to compute velocities and accel accurately (2**-16)
```

```
--
type RATE_TYPE is delta 1.525879E-5 range -32_768.0..32_767.0;
  for RATE_TYPE'size use 32;

sqrt_large_number : constant := 2508.0;  -- approx sqrt(LONG_FIXED'last)/4

type POSITION_TYPE is record            -- for absolute position
  X          : LONG_FIXED;              -- assume battlefield oriented ENU
  Y          : LONG_FIXED;
  Z          : LONG_FIXED;
end record;

type BAM is range -32768 .. 32767;      -- binary angle measurement 32768/180
                                        -- East North Up origins (0)
type EXTENDED_BAM is new LONG_INTEGER;  -- for large calculations

type AIMPOINT_TYPE is record
  AZIMUTH   : BAM;
  ELEVATION : BAM;
end record;

--
--  T80  - Main Battle Tank
--  SA9  - GASKIN surface to air missle launcher
--  BMP2 - Infantry Combat Vehicle
type TARGET_CLASS_TYPE is (UNKNOWN, T80, SA9, BMP2);

end Types;
```

```
-------------------------------------------------------------------
--% UNIT:     Distrib Package Body.                               --
-------------------------------------------------------------------
--|
--| PACKAGE BODY : Distrib
--|
--| OPERATION :
--| This package body makes calls to the runtime in order to obtain
--| configuration values which are based on the number of available
--| processors.
--|

-- Modifications Log
--
-- 88-12-05 : TEG => Original Created.
-- 89-12-06 : TEG => Enhanced to support dynamic configuration/reconfiguration
--


--------------------------------------------------
--  DISTRIBUTION CONTROL PARAMETERS             --
--------------------------------------------------
package body Distrib is

  type BOUND_TYPE is (LOW,HIGH);
  subtype GUIDE_RANGE is Types.WORD_INDEX range 1..Distrib.max_guide_tasks;
  ROCKET_CONFIG : array (GUIDE_RANGE,GUIDE_RANGE,BOUND_TYPE)
     of Types.WORD_INDEX :=
 -- if 1 task, all rockets on #1
     (1 => (1 => (LOW => 1, HIGH =>  20), 2 => (LOW => 1, HIGH =>  1)),
 -- if 2 tasks, 5 rockets on #1, 15 on #2
      2 => (1 => (LOW => 1, HIGH =>   5), 2 => (LOW => 6, HIGH => 20)));
---------------------------------------------------------------------
--
--  The follwing four functions provide configuration information based
--  on operator entered information and system configuration operations.
--  They are provided by the Distributed RunTime Environment
--
  function Get_Num_Rockets return Types.WORD_INDEX;

  function Get_Num_Targets return Types.WORD_INDEX;

  function Get_Num_Guide_Tasks return Types.WORD_INDEX;

  function Get_Master_Status return BOOLEAN;
-------------------------------------------------------------------
--   LATER DECLARATIVE ITEMS (BODIES)
--
--  RESTART is used to stop operation of the BDS and allow the operator
--  setup a different configuration.  It is only called when the MODE
--  button is pressed while the RESET button is held down on the mouse.
--  The Ada body version simply locks up the machine with interrupts disabled.
```

```
--
  pragma INTERFACE(ASM86, Get_Num_Guide_Tasks);
  pragma INTERFACE_SPELLING(Get_Num_Guide_Tasks, "D1DRTE?GETTASKS");

  pragma INTERFACE(ASM86, Get_Num_Targets);
  pragma INTERFACE_SPELLING(Get_Num_Targets, "D1DRTE?GETTARGETS");

  pragma INTERFACE(ASM86, Get_Num_Rockets);
  pragma INTERFACE_SPELLING(Get_Num_Rockets, "D1DRTE?GETROCKETS");

  pragma INTERFACE(ASM86, Get_Master_Status);
  pragma INTERFACE_SPELLING(Get_Master_Status, "D1DRTE?GETMASTER");
begin
  NUM_ROCKETS :=  Get_Num_Rockets;
  NUM_TARGETS :=  Get_Num_Targets;
  NUM_GUIDE_TASKS := Get_Num_Guide_Tasks;
  MASTER := Get_Master_Status;
  for I in Types.WORD_INDEX range 1..NUM_GUIDE_TASKS loop
    Guide_Low (I) := ROCKET_CONFIG(NUM_GUIDE_TASKS,I,LOW);
    Guide_High(I) := ROCKET_CONFIG(NUM_GUIDE_TASKS,I,HIGH);
  end loop;
end Distrib;
```

12 Appendix B - Distributed Runtime Source Code

The source code for the distributed runtime uses an 8086 family assembly language code. It is divided into modules which implement the major functional areas. These include: Initialization and system configuration, interprocessor synchronization, runtime routines, network setup, network I/O, distributed task control blocks, and the vendor runtime interface. Two include files: DA_HW.ASM and DA_DEF.ASM are used to define system constants and data structures.

```
        .XLIST
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   FILE:   DA_DEF.ASM                                                  ;
;   Distributed Ada - Definitions                                       ;
;                                                                       ;
;   Definitions for system values                                       ;
;                                                                       ;
;   Copyright (C) 1989, LabTek Corporation                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DEF_VRTIF_ADDR          equ     4000H
DEF_addr_size           equ     3       ; # of WORDs in Ethernet Address
;
; NETWORK MESSAGE CONTROL FIELD VALUES
; The first 6 fields are constant for ALL network traffic
;
packet                  struc
DEF_pkt_dest            dw      3 dup (?)
DEF_pkt_source          dw      3 dup (?)
DEF_pkt_length          dw      ?
DEF_pkt_sequence        dw      ?
DEF_pkt_cmd             dw      ?      ; designate type of message
DEF_pkt_TID             dw      ?      ; destinat'-- task id
DEF_pkt_entry_ID        dw      ?
DEF_pkt_my_PID          dw      ?      ; source processor ID
DEF_pkt_my_TID          dw      ?      ; source task ID
DEF_pkt_data            dw      ?      ; data always starts here
packet                  ends
;
;  Offset from list pointer to next node pointers
;
DEF_next_ptr            equ     2       ; offset to next pointer in buffer

DIR_entry       struc
DTCB_dir_local  dw      ?               ; local/distrib runtime flag
DTCB_dir_pid    dw      ?               ; PID for this task
DTCB_dir_TCB    dw      ?               ; pointer to distrib TCB
DTCB_dir_COUNT  dw      ?               ; Counter for task type
DIR_entry       ends

DTCB_dir_size   equ     size DIR_ENTRY              ; size of each entry


;
;   TCB Offsets
;
DEF_tcb_reply           equ     6
DEF_return_addr         equ     8
DEF_num_entries         equ     12
DEF_entry_table         equ     14
;
;   Within each TCB is an entry table
;   The table contains a record for each entry with the following fields:
```

```
;
DEF_entry_rec           struc
DEF_entry_profile_ptr   dw      ?
DEF_entry_wait          dw      ?
DEF_entry_queue         dw      ?
                        dw      ?
DEF_entry_rec           ends

;
;  CPU Designations
;
DEF_max_cpus            equ     3           ; maximum number of CPUs

DEF_alpha               equ     0
DEF_bravo               equ     1
DEF_charlie             equ     2
DEF_NA                  equ     -1          ; not applicable (no CPU)



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PROCESSOR / TASK / ENTRY IDs ;
;  Note: PIDs increment by 6,  ;
;        TIDs and EIDs by 2.   ;
; TASK IDs are unique.         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   COMMANDS received via messages
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DEF_sync_start          equ     0
DEF_sync_ready          equ     1
DEF_sync_continue       equ     2

DEF_request_entry       equ     3
DEF_rendezvous_end      equ     4

DEF_local_call          equ     5

DEF_ACK                 equ     -1

DEF_cold_start          equ     6

;
;   SYNC PHASE packet retry/delay values
;
DEF_retry_times         equ     5
DEF_sync_delay          equ     10              ; some delay between retries
DEF_WATCH_DOG_LIMIT     equ     100             ; 10ms per count

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;  Parameter Passing convention to runtime network msg routines.
;
; Standard Call Frame for IO_Xmit  (This is reverse order of being pushed)
; Therefore these values can be used relative to the BP
;
xmit            struc
                dw      2 dup(?); reserve space for near return and bp
DEF_PID         dw      ?       ; destination processor ID
DEF_CMD         dw      ?       ; command for this packet
DEF_TID         dw      ?       ; Task for which the command operates
DEF_ENTRY       dw      ?       ; entry ID for the command (if applicable)
DEF_MY_TID      dw      ?       ; originating Task ID
DEF_PROFILE     dw      ?       ; profile pointer (in CS) for entry parameters
DEF_MODE        dw      ?       ; current calling mode (in or out)
DEF_PARM_LIST   dd      ?       ; pointer (seg/offset) for parameter list
xmit            ends

DEF_xmit_frame  equ     size xmit      ; size of parameter frame

;
; Parameter Constraint Layout
;
constraint      struc
DEF_low_desc    dw      ?
DEF_high_desc   dw      ?
DEF_size_desc   dw      ?
constraint      ends

DEF_in          equ     1
DEF_out         equ     2
DEF_in_out      equ     3               ; bit-wise "or" of "in" & "out"

;
; Parameter Profile Layout
;
;
;        *************************
;        * Number of Parameters *
;        *************************
;        *   PARM1 : Mode       *     ; in, out, or in_out
;        *************************
;        *   PARM1 : Type/Length *    ; negative if unconstrained, otherwise
;        *************************    ; this is a WORD count
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                        ;
; Task Control Block Layout                                              ;
;                                                                        ;
; TASK_ID:      each block is pointed to by an entry in the              ;
;               TASK_DIRECTORY which is indexed by the TID.  The TID     ;
```

```
;              is essentially the task's priority (with a provision    ;
;              for tasks of the same type to have sequentially lower   ;
;              priority as they are created.                           ;
;                                                                      ;
;  Sync_Semaphore: The sync semaphore is used to suspend (or resume)   ;
;              execution of the associated task for rendezvous.        ;
;                                                                      ;
;  Reply Pointer: Contains the buffer descriptor of the reply msg.     ;
;                                                                      ;
;  Number of Entries: provides the number of entries for this task.    ;
;                                                                      ;
;  Entry_Table: The Entry table provides a record for each of the      ;
;              entries defined in the task.  The record contains:      ;
;                                                                      ;
;                      PROFILE_PTR: pointer to the parameter profile   ;
;                                described above.                      ;
;                      WAITING : flag indicating that the accepting    ;
;                                task is waiting for an entry call     ;
;                                for this entry.                       ;
;                      Queue   : Head of buffer descriptor linked to   ;
;                                this entry.                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        .LIST
```

```
        .XLIST
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   FILE:  DA_HW.ASM                                                   ;
;   Distributed Ada - Hardware Definition Include File                 ;
;                                                                      ;
;   Copyright(C) 1989, LabTek Corporation, Woodbridge, CT     USA      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                      ;
;  Ethernet Board Hardware Configuration
;                                                                      ;
base                        equ      310H      ; base address of board
vector_number               equ        5H      ; vector number for board
net_memory_seg              equ     0DC00H     ; address of ethernet memory
net_memory_size             equ      2000H     ; 8K bytes

;
;   LAN Controller Page 0 registers
;

NIC_cr          equ  base + 0;           -- control register of NIC
NIC_pstart      equ  base + 1;           -- page start register
NIC_pstop       equ  base + 2;           -- page stop register
NIC_bndy        equ  base + 3;           -- boundary register
NIC_tpsr        equ  base + 4;           -- transmit page start register
NIC_tbcr0       equ  base + 5;           -- transmit byte count rgtr hi
NIC_tbcr1       equ  base + 6;           -- transmit byte count rgtr lo
NIC_isr         equ  base + 7;           -- interrupt status register
NIC_rsar0       equ  base + 8;           -- remote start address rgtr lo
NIC_rsar1       equ  base + 9;           -- remote start address rgtr hi
NIC_rbcr0       equ  base + 10;          -- remote byte count rgtr lo
NIC_rbcr1       equ  base + 11;          -- remote byte count rgtr hi
NIC_rcr         equ  base + 12;          -- receive configuration rgtr
NIC_tcr         equ  base + 13;          -- transmit configuration rgtr
NIC_dcr         equ  base + 14;          -- data configuration register
NIC_imr         equ  base + 15;          -- interrrupt mask register
;
; controller page 1 registers - NIC address setup registers
; These registers are written to establish what the actual
; physical address will be.
;
phys_address_0 equ base + 1;      ; physical address registers.
phys_address_1 equ base + 2;      ; These registers are accessed
phys_address_2 equ base + 3;      ; via NIC_cr bits 7,6 = 0,1.
phys_address_3 equ base + 4;      ; LAN registers are accessed
phys_address_4 equ base + 5;      ; via cntrl bits 3,2 = 0,0.
phys_address_5 equ base + 6;      ;
NIC_curr       equ base + 7;      ; only written once during init

;
;   Controller Page 2 - Ethernet PROM ADDRESS memory
;   These locations contain the "preferred" address as contained
```

```
;   in PROM.  These will typically be copied to the physical
;   address registers above (page 1).
;
prom_address_0 equ  base + 0;         -- station address 0
prom_address_1 equ  base + 1;         -- station address 1
prom_address_2 equ  base + 2;         -- station address 2
prom_address_3 equ  base + 3;         -- station address 3
prom_address_4 equ  base + 4;         -- station address 4
prom_address_5 equ  base + 5;         -- station address 5

;
;   Gate Array registers (note: offset of 400H)
;

pstr           equ  base + 400H;     -- page start register
pspr           equ  base + 401H;     -- page stop register
dqtr           equ  base + 402H;     -- drq timer register
bcfr           equ  base + 403H;     -- base configuration register
pcfr           equ  base + 404H;     -- prom configuration register
gacfr          equ  base + 405H;     -- ga configuration register
cntrl          equ  base + 406H;     -- gate array (ga) control rgtr
streg          equ  base + 407H;     -- ga status register
idcfr          equ  base + 408H;     -- interrupt/DMA cnfgrtn rgtr
damsb          equ  base + 409H;     -- DMA address register hi
dalsb          equ  base + 40AH;     -- DMA address register lo
vptr2          equ  base + 40BH;     -- vector pointer rgtr #2
vptr1          equ  base + 40CH;     -- vector pointer rgtr #1
vptr0          equ  base + 40DH;     -- vector pointer rgtr #0
rfmsb          equ  base + 40EH;     -- register file access hi
rflsb          equ  base + 40FH;     -- register file access lo

;******************************************
;* Ethernet (3com) Initialization Values *
;******************************************

eth_enable_reset     equ   03h          ; enable reset
eth_disable_reset    equ   00h          ; disable reset
eth_access_prom      equ   04h          ; access prom bytes
eth_recv_select      equ   00h          ; select external Xceiver
eth_lan_config       equ   49h          ; 8k of mem-map I/O, w/interrupts
eth_rem_DMA_burst    equ   08h          ; # of bytes to transfer on DMA burst
eth_irq_line         equ   80h          ; interrupts occur on IRQ5
eth_rem_DMA_config   equ   20h          ; 8k configuration for remote DMA
eth_xmit_buf_start   equ   20h          ; begin of transmission buffer (0H)
eth_recv_buf_start   equ   26h          ; receive queue            (0600H)
eth_recv_buf_end     equ   40h          ; 20 pages, 256 bytes/page  (2000H)
eth_offset           equ   2000h        ; difference between page & address
eth_recv_begin       equ   600h         ; actual offset in RAM seg for begin
eth_recv_end         equ   2000h        ; actual offset in RAM seg for end
eth_start_nic        equ   02h          ; start NIC
eth_nic_stop         equ   01h          ; stop the NIC
```

```
eth_nic_DMA_config  equ   48h           ; local DMA operations, 8 byte bursts
eth_remote_DMA_lo   equ   00h           ; DMA remote unused (lo)
eth_remote_DMA_hi   equ   00h           ; DMA remote unused (hi)
eth_packet_types    equ   00h           ; receive only good packets
eth_nic_mode        equ   02h           ; internal loopback mode
eth_bndy_start      equ   00h           ; FOR NOW, DO NOT USE BOUNDRY REG!
eth_int_status      equ   0ffh          ; clear status of all ints at start
eth_ints_disabled   equ   00h           ; enable no interrupts
eth_access_page_0   equ   00h           ; access page 0 again  (for cmd reg)
eth_access_page_1   equ   40H           ; access NIC page 1 registers
eth_exit_mode       equ   00h           ; exit internal loopback mode

nic_prx             equ     1           ; mask for packet receive interrupt
nic_ptx             equ     2           ; mask for packet transmit interrupt

send                equ     4           ; command byte to start transmission

;
;  Interrupt Controller Commands
;
NET_EOI        equ  60H + vector_number  ; -- End Of Interrupt (specific)
TIMER_EOI      equ  60H + 0              ; timer is interrupt channel 0

; Ethernet controller routine specifications
;
; Ethnet_Init initializes a 3com Etherlink II board to transmit and receive
; packets via a memory mapped interface with the board located at DC00:0000.
; The base address from which the registers are located is 310h.  The init
; routine intializes the memory to zeroes before it completes.  Although no
; DMA is used to transfer the data from main memory to the board's memory
; (which is referred as remote DMA operations), there is no choice but to
; use the local DMA operations (transferring bytes or words from the board's
; memory to the board's output fifo's).
        .LIST
```

```
        page    55,132
        TITLE   RTE - Distribted Ada Runtime Module
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA_RTE.ASM                                                 ;
;                                                                    ;
;  RTE -  DISTRIBUTED Ada RUNTIME MODULE                             ;
;                                                                    ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT    USA      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                    ;
;  Runtime Code to implement prototype Distributed Ada Services      ;
;                                                                    ;
;  This module implements the remote rendezvous operations to        ;
;  support distributed Ada.                                          ;
;                                                                    ;
;  Currently provided are:                                           ;
;    Remote_Entry, Remote_Select, Remote_Accept, Remote_End_Accept,  ;
;    Remote_Elab_Start, Remote_Elab_Wait, Remote_Elab_Continue.      ;
;    Local_End_Accept                                                ;
;                                                                    ;
;  Ver Date     Description                                          ;
;                                                                    ;
;  0.1 Nov-88 : Initial prototype                                    ;
;  0.2 Dec-89 : Version 2 - Flexible task distribution Added         ;
;                                                                    ;
;                                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        .model  large

        public  Initialize
;
;   These are entry points called by the vendor runtime interface to
;   invoke the runtime by generated code
;
        public  Request_Entry, Activate_Complete, Accept, Rendezvous_Complete
        public  Select, Create_Task
;
;   The IO module invokes the runtime services when messages are received
;   via the NET_RECEIVE call
;
        public  NET_Receive             ; called by IO


; Vendor Runtime Services
        extrn   VRTIF_Init:near
        extrn   VRTIF_Wait:far          ; Vendor Supplied P Semaphore operation
        extrn   VRTIF_Signal_I:far      ; Vendor Supplied V operation/interrupt
        extrn   VRTIF_Signal:far        ; Vendor Supplied V operation
;
;  After or instead of using the distributed runtime, control may be
;  passed bact to the vendor runtime through this interface
```

```
;
        extrn   VRTIF_Create_Task:near
        extrn   VRTIF_Activate_Complete:near
        extrn   VRTIF_Entry:near
        extrn   VRTIF_Rendezvous_Complete:near
        extrn   VRTIF_Accept:near
        extrn   VRTIF_Select:near
        extrn   VRTIF_Lower_Priority:near
;
;  Vendor task control block information and runtime data segment address
;
        extrn   VRTIF_tcbtid:abs        ; offset to priority within vendor TCB
        extrn   VRTIF_task_ptr:word     ; offset to current TCB with runtime DS
        extrn   VRTIF_DS:word           ; offset within user DS to runtime DS
        extrn   VRTIF_SELECT_REC:abs    ; number of bytes per "select record"

        extrn   Sync:near               ; call synchronize
        extrn   Shut_Down:near          ; restart system on "COLD_START"
        extrn   TASK_DIRECTORY:word

; Network IO Services
        extrn   TX_READY:near           ; Transmit ready semaphore
        extrn   IO_XMIT:near            ; Start transmission routine
        extrn   IO_Network_Init:near
        extrn   IO_ALLOCATE:near        ; allocate a buffer
        extrn   IO_DEALLOCATE:near      ; deallocate a buffer
        extrn   PID:word                ; THIS processor ID

        extrn   SYNCHRO_SEMAPHORE:word
        extrn   CONTINUE_SEMAPHORE:word

        extrn   Outchr:near             ; for debugging only

        include DA_DEF.ASM              ; system definitions


cseg    segment common
        assume  cs:cseg,ds:cseg,es:cseg
        org     1400H
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Initialize    -- no parameters
;
Initialize:
        call    IO_Network_Init
        call    VRTIF_init
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Prior to each Create Task, synchronize all CPU's to keep elaboration
;  going sequencially
```

```
;
Create_Task:
        push    ds
        push    ax
;        mov     al,'c'
;        call    Outchr          ; @@
        mov     ax,cs
        mov     ds,ax
        call    Sync            ; do synchronization
        pop     ax
        pop     ds
        jmp     VRTIF_Create_Task ; return to vendor runtime

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  A task has completed activation and called "ACTIVATED". Since there
;  are no parameters, simply nest the call to the vendor runtime so it
;  will return here when done. First, provide a unique ID based on
;  priority for each task.  Then we see if the task should remain
;  alive. If not, suspend it on a dummy semaphore
;
Activate_Complete:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es
        mov     ds,[VRTIF_DS]
        mov     si,[VRTIF_TASK_PTR]
        mov     cx,[si+VRTIF_TCBTID]    ; get priority (of this task type)
;
;  During Activation all tasks have the priority of their task type, however
;  since the priority is used to identify tasks, and possibly serveral tasks
;  will be of the same task type, count the tasks of each task type and
;  assign them a unique priority.  (note the initial priorities must be
;  assigned with sufficient space so that this has no effect on scheduling).
;  Decreasing Ada priorities have increasing VRT priority (by two).
;
        mov     di,cx
        add     di,di                   ; mult by four to make index
        add     di,di
;
;   Modifying count for base task is atomic action
;
        pushf
        cli
```

```
        mov     ax,cs:TASK_DIRECTORY.DTCB_dir_Count[di]; get # of tasks for type
        add word ptr cs:TASK_DIRECTORY.DTCB_dir_Count[di],2 ; adjust it for next
        popf

        or      ax,ax                   ; see if delta on this priority
        jz      ACT_COMPLETE10          ; if so skip changing of pririty
        add     cx,ax                   ; compute new priority
        push    cx                      ; save priority
        call    VRTIF_Lower_Priority    ; setpriority lower cx=priority si=V-TCB
        pop     di
        add     di,di
        add     di,di
ACT_COMPLETE10:
        mov     ax,cs                   ; set DRT data segment
        mov     ds,ax

        mov     bx,TASK_DIRECTORY.DTCB_dir_TCB[di]      ; get DRT TCB
        lea     bx,DEF_return_addr[bx]                  ; point to return addr

        mov     di,[bp+2]               ; get return address offset
        mov     [bx],di                 ; save in TCB
        mov     di,[bp+4]               ; get return address segment
        mov     [bx+2],di

        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     bp
        add     sp,4                    ; trash return address (saved in TCB)

        push    cs                      ; simulate a FAR call
        call    VRTIF_Activate_complete

        sub     sp,4                    ; make room for return address
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        push    cx
        push    dx                      ; "mul" affects this
        push    si
        push    di
        push    ds
        push    es

        mov     ds,[VRTIF_DS]           ; get Vendor runtime data segment
```

```
        mov     si,[VRTIF_TASK_PTR]      ; fetch Current Task TCB
        mov     bx,[si+VRTIF_TCBTID]     ; get priority (our task type)
        mov     ax,cs                    ; load DRT data segment
        mov     ds,ax
        add     bx,bx                    ; mult by four to make index
        add     bx,bx
        mov     ax,TASK_DIRECTORY.DTCB_dir_pid[bx]; fetch PID for this task
        cmp     ax,[PID]                 ; see if this is the processor
        jz      Keep_alive
;
;  If here, this task should not continue to run... suspend it.
;
        pushf                            ; this must be atomic
        cli
        xor     ax,ax                    ; init a dummy semaphore
        mov     [DUMMY_SEM],ax
        mov     [DUMMY_SEM+2],ax
        mov     [DUMMY_SEM+4],ax
        push    cs
        lea     ax,DUMMY_SEM
        push    ax
        call    VRTIF_Wait               ; go to sleep forever
        popf
        int     3                        ; if here... ERROR!
;
;  This task should be allowed to live, let it continue
;
Keep_alive:
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[bx]       ; fetch TCB
        lea     si,DEF_return_addr[si]
        mov     ax,[si]                  ; fetch offset
        mov     [bp+2],ax                ; put on stack
        mov     ax,[si+2]                ; fetch segment
        mov     [bp+4],ax

        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     bp
        retf


;
;   Programs will come here when they want to do an entry call.  If
;   the call is to a task with remote callers, we must go through the
;   distributed runtime, otherwise go to local runtime.
```

```
;
;   Input parameters:
;          ES:DX points to Parameter List
;          BX    task id to call
;          CX    entry id to call
;
Request_Entry:
        push    ds              ; save for vendor runtime
        mov     ds,[VRTIF_DS]   ; fetch local runtime data segment
        mov     si,[bx+VRTIF_tcbtid] ; get distributed task id (priority)
        mov     ax,si           ; save task id
        mov     di,[VRTIF_TASK_PTR]  ; get vendor TCB of current task
        mov     di,[di+VRTIF_tcbtid] ; fetch distrib. tid of current task
        pop     ds              ; restore ds
        add     si,si           ; mult by four to make index
        add     si,si
        test    cs:TASK_DIRECTORY[si],0FFFFH     ; see if local or remote
        jnz     Dist_Entry      ; if entry must be done by distributed runtime
;
;  This entry is strickly local, let vendor runtime handle it
;
        jmp     VRTIF_Entry     ; go to vendor runtime

;
;  This entry must be handled by the distributed runtime, check to see if
;  the called task is on this processor.
;
Dist_Entry:
;        push    ax
;        mov     al,'r'
;        call    Outchr
;        pop     ax
        mov     si,cs:TASK_DIRECTORY.DTCB_dir_pid[si] ; fetch PID of called task
        cmp     si,cs:[PID]                        ; compare against my PID
        jnz     Remote_Entry
        jmp     Local_Entry                     ; do local if the same

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Remote_Entry
;
;   Send "Request_Entry" message with copied parameters and profile
;   Wait on Entry_Wait_Semaphore
;   Copy OUT parameters
;   Release Buffer
;
;   IN PARAMETERS:
;          AX    distributed task id to call
;          CX    entry id to call
;          DI    distributed task ID of this caller task
;          ES:DX points to Parameter List
```

```
;
;   NOTE: Stack Parameters are removed by caller
;
RE_Parm_List    equ     -4      ; dword
RE_Profile      equ     -6      ; parameter profile ptr
RE_Count        equ     -8      ; parameter count
RE_TCB          equ     -10     ; dist. TCB of server
RE_TID          equ     -12     ; Distrib. Task ID of caller
RE_BUFF_DESC    equ     -14     ; descriptor of reply buffer
RE_BUFF_PTR     equ     -16     ; address to packet data

Remote_Entry:
;        push    ax
;        mov     al,'R'
;        call    Outchr
;        pop     ax
        push    bp
        mov     bp,sp
        add     sp,RE_BUFF_PTR          ; local parameters to save
        push    ds                      ; save caller's  DS
        push    cs
        pop     ds                      ; load DRT data segment
        mov     [bp+RE_TID],di          ; save task id of caller
        mov     [bp+RE_Parm_list+2],es  ; save segment of parameter list
        mov     [bp+RE_Parm_list],dx    ; save offset of parameter list
;
;  Build call frame to transmit entry call to designated task
;
;  PARM_LIST
        push    es              ; push segment of parameter list
        push    dx              ; push offset of parameter list
        mov     dx,DEF_in       ; calling Xmit for IN mode
;  MODE
        push    dx
        mov     bx,cx           ; get entry id
        add     bx,bx           ; mult by two
        add     bx,bx           ; mult by four
        add     bx,bx           ; mult by eight (8 bytes per entry descriptor)
        mov     si,ax           ; get destination TID in SI
        add     si,si           ; mult by four to make index
        add     si,si
        mov     si,Task_Directory.DTCB_dir_TCB[si]      ; fetch dist. TCB
        mov     [bp+RE_TCB],si  ; save for later
        mov     dx,DEF_Entry_Table+DEF_entry_profile_ptr[si+bx] ;profile offset
        mov     [bp+RE_Profile],dx   ; save for later
;  PROFILE
        push    dx              ; push as parameter
;  TID of Source
        push    di
;  ENTRY
        push    cx              ; push entry id
```

```
;  TID of Destination
        push    ax              ; push (DA) task id
;  CMD for remote entry call
        mov     di,DEF_request_entry
        push    di
;  PID of Destination
        mov     di,ax           ; get back TID of dest.
        add     di,di           ; mult by four to make index
        add     di,di
        mov     di,Task_Directory.DTCB_dir_PID[di]   ; fetch PID
        push    di              ; push PID

        call    IO_Xmit         ; parameters are copied by xmit
;
;   Now wait for rendezvous Complete to wake up
;
        mov     si,[bp+RE_TID]          ; get my dist TID
        add     si,si
        add     si,si
        mov     si,Task_Directory.DTCB_dir_TCB[si]   ; fetch my dist. TCB
        push    cs
        push    si              ; base of DA TCB is semaphc e
        call    VRTIF_Wait      ; go to sleep waiting for end rendezvous
;
;   Copy out parameters back.  Use the TCB definitions
;   to determine how many parameters, their size, and what type (ie. must
;   allow for unconstrained arrays).
;
;
;  First get address of buffer and stick it in local
;
        mov     si,[bp+RE_TID]          ; get my dist TID
        add     si,si
        add     si,si
        mov     si,Task_Directory.DTCB_dir_TCB[si]   ; fetch dist. TCB
        mov     si,DEF_TCB_Reply[si]    ; get reply buffer descriptor
        mov     [bp+RE_BUFF_DESC],si    ; save it for later deallocation
        mov     si,[si]                 ; get actual buffer address
        lea     si,DEF_pkt_data[si]     ; point directly to data
        mov     [bp+RE_BUFF_PTR],si     ; save pointer

        mcv     si,[bp+RE_Profile]      ; get parameter profile ptr
        cld                             ; make sure auto increment
        lodsw                           ; get number of parameters

R_Entry_10:
        or      ax,ax                   ; see if done
        jnz     R_Entry_12              ; continue if not done
        jmp     R_Entry_30              ; if done
R_Entry_12:
        mov     [bp+RE_COUNT],ax        ; update parameter count
```

```
        lodsw                           ; get parameter Mode
        mov     cx,[si]                 ; fetch parameter type/length
        add     si,2                    ; skip over type/length
        mov     [bp+RE_PROFILE],si      ; update profile pointer for next

        lds     si,[bp+RE_parm_list]    ; point to parameter list
; note: vendor puts segment/offset in reverse of normal order
        push    data_seg[si]            ; segment of data
        push    data_off[si]            ; offset of data

        or      cx,cx                   ; see if unconstrained type
        jge     R_Entry_15
;
; process an unconstrained object as a parameter.  Note, the
; descriptor is always copied, so we must skip 3 words in buffer
; and over two in the parameter list
;
        push    [si+4]                  ; descriptor segment
        push    [si+6]                  ; offset of descriptor
        add     si,8
        mov     word ptr [bp+RE_parm_list],si ; update parameter list index
        mov     si,[bp+RE_BUFF_PTR]     ; adjust buffer pointer over constraint
        add     si,6                    ; skip over constraint
        mov     [bp+RE_BUFF_PTR],si     ; update
        pop     si                      ; get offset of descriptor
        pop     ds                      ; get segment of descriptor
        push    ax                      ; save MODE of parameter
        mov     cx,[si+DEF_low_desc]    ; get low bound of constraint
        mov     ax,[si+DEF_high_desc]   ; get high bound of constraint
        mov     dx,[si+DEF_size_desc]   ; get size of object
;
; Copy the parameter data iff MODE is correct and array is not null
;
        mov     bx,cs                   ; reload DRT data segment
        mov     ds,bx
        pop     bx                      ; get mode of parameter
        pop     di                      ; get offset of data
        pop     es                      ; get segment of data
        and     bx,DEF_out              ; see if we should copy data
        jz      R_Entry_20              ; if not, go on
        sub     ax,cx                   ; compute difference in range
        inc     ax                      ; adjust to include end points
        mul     dx                      ; compute size in words
        jle     R_Entry_20              ; if array is empty go to next parm
        mov     cx,ax                   ; put in count register
        mov     si,[bp+RE_BUFF_PTR]
        rep     movsw                   ; transfer from packet buffer
        mov     [bp+RE_BUFF_PTR],si     ; update pointer
        jmp     R_Entry_20              ; go on to next parameter
;
;  Constrained parameter, CX is length in bytes, copy it into packet buffer
```

```
;
R_Entry_15:
        add     si,4                    ; move to next object address
        mov     word ptr [bp+DEF_parm_list],si ; update parameter list index
        pop     di                      ; get data offset
        pop     es                      ; get parameter data segment
        and     ax,DEF_out              ; see if mode is right to copy out
        mov     ax,cs                   ; restore distrib. data segment
        mov     ds,ax
        jz      R_Entry_20              ; skip copy of data if not out mode
        mov     si,[bp+RE_BUFF_PTR]     ; get buffer pointer in DS:SI
        inc     cx                      ; round odd bytes up when convert
        shr     cx,1                    ; to words
        rep     movsw
        mov     [bp+RE_BUFF_PTR],si     ; update current packet buffer ptr
R_Entry_20:
        mov     si,[bp+RE_Profile]      ; get next parameter profile
        mov     ax,[bp+RE_Count]        ; get the counter back in ax
        dec     ax                      ; count down
        jmp     R_Entry_10
;
;  Free buffer, restore stack, and return to entry caller
;  (it restores DS and any any stack frame it may have built)
;
R_Entry_30:
        mcv     bx,[bp+RE_BUFF_DESC]    ; get reply buffer descriptor back
        call    IO_Deallocate           ; return used buffer
        pop     ds                      ; restore caller's data segment
        mov     sp,bp                   ; dallocate locals
        pop     bp
        retf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Local_Entry  : This routine is called for an entry of a task
;                   which is local (same processor) as the caller
;
;     Inputs:
;       AX : TID of called task
;       BX : Vendor TCB of called task
;       CX : Entry ID
;       DX : offset to parmeter list
;       ES : segment of parameter list
;       si : PID of called task
;       di : MY_TID
;
;   Although the task is local to the caller, an IO buffer is allocated
;   to store the necessary pointers required by accepting tasks.  This
;   is later deallocated as part of the local_end_accept routine.  The
;   calling task is always suspended, and if the accepting task is "waiting"
;   it is signaled to wake up.
```

```
;
;   Only the TID EID and MY_TID fields within the buffer are valid during
;   local rendezvous.  Also, the data fields have the address of the
;   various objects/desciptors rather than the data itself.
;
;   NOTE: There is no need to deallocate the buffer allocated here because
;         it is deallocated by the server task.  (There is only one
;         buffer used by local tasks, rather than two as for remote tasks.)

LE_ENTRY_PTR     equ     -2              ; word: bp offset to current entry table
LE_TCB_PTR       equ     -4              ; word: bp offset to target TCB base
LE_MY_TCB        equ     -6              ; word: bp offset to my TCB base


Local_Entry:
;         push    ax
;         mov     al,'L'
;         call    Outchr
;         pop     ax
        push    bp
        mov     bp,sp
        add     sp,LE_MY_TCB            ; allocate space for locals
        push    ds                      ; save caller's data segment
        push    cs                      ; load DRT data segment
        pop     ds
        push    ax                      ; save TID
        call    IO_Allocate             ; get a buffer descriptor ptr in BX
        pop     ax
        mov     si,[BX]                 ; fetch buffer address
;
;   currently only one parameter is used (either in or out).  Take advantage
;   of this to simplify interface to accepting task.  The address of the
;   data area is provided in the first part of the buffer.  NOTE: this address
;   is backwards (segment=low address, offset=high address).
;
        push    ax
        push    di
        mov     di,dx
        mov     ax,es:[di]              ; transfer parm list to buffer
        mov     [si],ax                 ; buffer so as to point to the
        mov     ax,es:[di+2]              ; data and descriptors actually
        mov     [si+2],ax               ; processor
        mov     ax,es:[di+4]
        mov     [si+4],ax
        mov     ax,es:[di+6]
        mov     [si+6],ax
        pop     di
        pop     ax

        mov     DEF_pkt_tid[si],ax       ; put in called task TID
        mov     DEF_pkt_my_tid[si],di    ; and put in calling task id there
```

```
        mov     DEF_pkt_cmd[si],DEF_local_call ; indicate this is a local call
        add     di,di                   ; mult by four
        add     di,di
        mov     di,TASK_DIRECTORY.DTCB_dir_TCB[di] ; get my TCB addr
        mov     [bp+LE_MY_TCB],di         ; save it

        mov     [si+DEF_pkt_Entry_ID],CX  ; save entry id
        mov     si,ax                     ; get TID of called task into si
        add     si,si                     ; mult by four
        add     si,si
        mov     ax,TASK_DIRECTORY.DTCB_dir_TCB[si]  ; fetch dist. TCB addr
        mov     [bp+LE_TCB_PTR],ax        ; save base of TCB
        mov     si,cx                     ; compute entry table address
        add     si,si                     ; * 2
        add     si,si                     ; * 4
        add     si,si                     ; * 8
        add     si,ax                     ; add base of (DA) TCB
        add     si,Def_Entry_table
        mov     [bp+LE_Entry_PTR],si      ; save
        lea     si,DEF_Entry_Queue[si]    ; fetch entry queue head
;
;  ATOMIC action follows...  Queue entry, if waiting signal acceptor
;
        pushf
        cli
        call    INSERT                  ; place buffer descriptor on entry Q
        mov     si,[bp+LE_Entry_PTR]    ; fetch entry table address again
        test    DEF_Entry_Wait[si],0FFFFH    ; see if WAITING
        jz      le020                   ; go on if not
;
;  server is waiting on accept, signal it
;
        mov     si,[bp+LE_TCB_PTR]      ; get task Control Block
        mov     cx,DEF_num_entries[si]  ; get number of entries
        lea     si,DEF_entry_table[si]  ; point to base of table
le010:
        mov     DEF_Entry_Wait[si],0    ; clear (all) waiting flags
        add     si,size DEF_Entry_Rec   ; go to next entry record
        loop    le010

        push    cs                      ; segment of semaphore
        mov     ax,[bp+LE_TCB_PTR]      ; offset of semaphore
        push    ax
        call    VRTIF_Signal            ; wake up server (may preempt ourselves)
;
;  NOTE: This is the end of the atomic region (above Vendor runtime call
;        reenables interrupts!
;
le020:
        popf                            ; restore interrupt level
        push    cs                      ; now try to suspend ourselves
```

```
        mov     ax,[bp+LE_MY_TCB]       ; semaphore is first thing in TCB
        push    ax
        call    VRTIF_Wait              ; may not suspend if server is higher
                                        ; priority and has already signaled us!

        pop     ds                      ; restore DS
        mov     sp,bp                   ; remove locals
        pop     bp
        retf



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Accept - is invoked by the generated code to wait for arrival of
;          a caller.
;               INPUTS:
;                       AX is entry to accept
;               OUTPUTS:
;                       ES:BX is parameter list pointer
Accept:
        push    ds
        mov     ds,[VRTIF_DS]           ; get runtime data segment
        mov     si,[VRTIF_TASK_PTR]
        mov     si,VRTIF_tcbtid[si]     ; fetch my TID
        pop     ds
        add     si,si                   ; mult by four to make index
        add     si,si
        test    cs:TASK_DIRECTORY.DTCB_dir_LOCAL[si],OFFFFH ; distributed?
        jnz     Dist_Accept             ; must to a distributed accept
        jmp     VRTIF_Accept            ; otherwise, return to vendor runtime
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Distributed Accept (TASK_ID, ENTRY_ID) return ES:BX_Param_Pointer
;
;    NOTE: THIS HANDLES BOTH ACCEPTANCE FOR LOCAL AND REMOTE CALLS
;          THROUGH THE DISTRIBUTED RUNTIME.
;
;    Simple Accept, see if someone on entry queue, if so
;    return with pointer to buffer in ES:BX, otherwise set
;    "Waiting" Flag and go to sleep on semaphore.
;
;    Inputs:  TASK_ID, ENTRY_ID
;
;    Outputs: Returns ES:BX pointing to Parameter Data List
;             Also, Buffer descriptor is placed in "Reply" pointer.
;
RA_TCB          equ     -2      ; word: my TCB
RA_ENTRY        equ     -4      ; word: this entry
```

```
Dist_Accept:
;        push    ax
;        mov     al,'a'
;        call    Outchr          ; @@
;        pop     ax
        push    bp
        mov     bp,sp
        sub     sp,4
        push    ds              ; save old data segment
        push    cs              ; load data segment
        pop     ds
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[si]      ; fetch TCB ptr
        mov     [bp+RA_TCB],si                          ; save it
        mov     bx,ax                       ; compute entry index
        add     bx,bx           ; * 2
        add     bx,bx           ; * 4
        add     bx,bx           ; * 8   (eight bytes per entry)
        lea     bx,DEF_entry_table[si+bx]       ; point to my entry of interest
        mov     [bp+RA_ENTRY],bx                ; save it too
        pushf                   ; save interrupt status
        cli                     ; go atomic
        test    DEF_entry_Queue+2[bx],0FFFFH ; if Zero, then queue is empty
        jnz     RA010           ; if caller is there, take it!
;
;  No caller on entry queue.  Set waiting flag and go to sleep
;
        mov     [bx+DEF_entry_wait],1 ; set flag
        push    cs              ; push segment of my task semaphore
        push    si              ; address of my tcb
        call    VRTIF_WAIT      ; go to sleep waiting for caller
;
; NOTE after vendor runtime call - interrupts are enabled!
;
; Now Something is on the queue, provide address of parameter list in
; ES:BX and return to caller.
;
RA010:
        popf                            ; restore interrupt status
        mov     si,[bp+RA_TCB]          ; get TCB pointer back
        mov     bx,[bp+RA_ENTRY]        ; get the entry address back
; note: the wait flag is cleared by the caller
        mov     bx,DEF_entry_queue+2[bx] ; get buffer descriptor from queue
        mov     DEF_tcb_reply[si],bx ; save descriptor for end rendezvous
        mov     bx,[bx]                 ; fetch buffer address into BX (return)
        mov     ax,cs                   ; get segment into ES, making ES:BX pair
        mov     es,ax                   ; parameter list is in buffer
                                        ; It has been put there by either the
                                        ; local or remote entry call mechanisms
        pop     ds                      ; restore data segment
        mov     sp,bp                   ; remove locals
        pop     bp
```

```
        retf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Select -
;       INPUTS:
;               STACK frame has open alternatives.  As best as we can
;               tell, it looks like this:
;                       [ flags ]
;                       [ entry # ]
;                       [ unknown ]
;               Each alternative appears to have three words with the
;               "flags" word being not-zero.  If it is zero, this indicates
;               the end of the list.
;
;       OUTPUTS:
;               All input parameters are removed from the stack and replaced
;               the parameter list pointer and a selector which indicates
;               which alternative was selected.
;
SELECT_LIST     equ     6               ; offset from bp to open alternatives
FLAGS           equ     0               ; offset to flags withing list record
ENTRY_ID        equ     2               ; offset to ID# within list record

Select:
        push    ds
        mov     ds,[VRTIF_DS]           ; get runtime data segment
        mov     si,[VRTIF_TASK_PTR]
        mov     si,VRTIF_tcbtid[si]     ; fetch my TID
        pop     ds
        add     si,si                   ; mult by four to make index
        add     si,si
        test    cs:TASK_DIRECTORY.DTCB_dir_local[si],0FFFFH ; distributed?
        jnz     Dist_Select             ; must to a distributed select
        jmp     VRTIF_Select            ; otherwise, return to vendor runtime

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Dist_Select
;
;   Check to see if any of the entries have callers.  If not,
;   set the "Waiting" Flag in each of them, and go to sleep.
;   If one entry has a queued request, accept it and return
;   offset for "Case" table and parameter list pointer on the stack
;   The offset for the case table is the entry id + 1.
;
;   INPUTS: Index into TASK_DIRECTORY is in SI
;
DS_TCB          equ     -2                  ; word: my TCB
DS_ENTRY        equ     -4                  ; word: this entry
DS_ALTER        equ     -6

Dist_Select:
```

```
;        mov     al,'s'
;        call    Outchr          ; @@
        push    bp
        mov     bp,sp
        add     sp,DS_ALTER             ; allocate local storage
        push    ds                      ; save DS
        mov     ax,cs
        mov     ds,ax                   ; set to Distr. runtime data segment
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[si]      ; fetch TCB ptr
        mov     [bp+DS_TCB],si                          ; save it
;
;   ENTER CRITICAL REGION (cannot allow task to go on an entry queue
;   after we have checked it, but before setting waiting flag.
;
        pushf
        cli
;
;  First check each entry to see if any has a caller...
;  Go through all open alternatives
;
Rem_Sel00:                              ; will come back here after resume
        lea     ax,[bp+C  . ~ _LIST]    ; get address of entry list
        mov     [bp+DS A  _R],ax        ; save in local variable
Rem_Sel10:
        mov     bx,[bp+DS_ALTER]        ; get pointer
        test    SS:FLAGS[bx],0ffffH     ; test if end of the list
        jz      Rem_Sel15               ; did not find it
        mov     ax,SS:ENTRY_ID[bx]      ; get entry ID
        mov     bx,ax                   ; compute entry index
        add     bx,bx                   ; * 2
        add     bx,bx                   ; * 4
        add     bx,bx                   ; * 8   (eight bytes per entry)
        lea     bx,DEF_entry_table[si+bx] ; point to entry of interest
        test    DEF_entry_queue+2[bx],0FFFFH ; if Zero, then queue is empty
        jnz     Rem_Sel50               ; if caller is there, take it!
        add     word ptr [bp+DS_ALTER],VRTIF_SELECT_REC  ; bytes per record
        jmp     Rem_Sel10               ; loop till end of list
;
;   all of the Entry Queues are Empty, mark each Waiting flag
;   and go to sleep.
;
Rem_Sel15:
        lea     ax,[bp+SELECT_LIST]     ; get address of entry list
        mov     [bp+DS_ALTER],ax        ; save in local variable
Rem_Sel20:
        mov     bx,[bp+DS_ALTER]        ; get pointer
        test    SS:FLAGS[bx],0ffffH     ; test if end of the list
        jz      Rem_Sel30               ; done
        mov     ax,SS:ENTRY_ID[bx]      ; get entry ID
        mov     bx,ax                   ; compute entry index
        add     bx,bx                   ; * 2
```

```
        add     bx,bx                   ; * 4
        add     bx,bx                   ; * 8   (eight bytes per entry)
        lea     bx,DEF_entry_table[si+bx] ; point to entry of interest
        mov     DEF_entry_wait[bx],1    ; set waiting
        add     word ptr [bp+DS_ALTER],VRTIF_SELECT_REC
        jmp     Rem_Sel20               ; loop till end of list

;
;   The following runtime call will suspend this task, when it
;   resumes, the interrupt flag will be set again, and presumably,
;   one of the entries will have a caller queued.
;
Rem_Sel30:
        push    cs                      ; push segment of wait_semaphore
        push    [bp+DS_TCB]             ; push offset  of wait_semaphore taskid
        call    VRTIF_Wait              ; do wait on semaphore
;
;   Now clear all the waiting flags
;
        cli
        lea     ax,[bp+SELECT_LIST]     ; get address of entry list
        mov     [bp+DS_ALTER],ax        ; save in local variable
Rem_Sel40:
        mov     bx,[bp+DS_ALTER]        ; get pointer
        test    SS:FLAGS[bx],0ffffH     ; test if end of the list
        jz      Rem_Sel45               ; done
        mov     ax,SS:ENTRY_ID[bx]          ; get entry ID
        mov     bx,ax                   ; compute entry index
        add     bx,bx                   ; * 2
        add     bx,bx                   ; * 4
        add     bx,bx                   ; * 8   (eight bytes per entry)
        lea     bx,DEF_entry_table[si+bx] ; point to entry of interest
        mov     DEF_entry_wait[bx],0    ; clear waiting
        add     word ptr [bp+DS_ALTER],VRTIF_SELECT_REC
        jmp     Rem_Sel40               ; loop till end of list

Rem_Sel45:
        jmp     Rem_Sel00               ; go back and find caller

;
;   There is a caller on this entry queue, do start accept
;   Fetch the Caller's buffer, which has a (backward) pointer to
;   the parameter data
;
Rem_Sel50:
        popf                            ; no longer critical
        mov     si,DEF_entry_queue+2[bx]  ; fetch buffer descriptor
        mov     di,[bp+DS_TCB]          ; get base of my TCB back
        mov     DEF_tcb_reply[di],si    ; put buff descriptor into reply ptr
        mov     si,[si]                 ; get actual buffer (which is parm list)
        inc     ax                      ; make entry id # compatible with VRTIF
```

```
;
;   Now pull parameters off of stack, and replace with parm_list ptr and
;   case selector
;
        pop     ds                      ; get DS back
        mov     sp,bp                   ; start with all locals
        pop     bp                      ; get back saved bp
        pop     bx                      ; get return offset
        pop     cx                      ; get return segment
;
;   Go thru open alternative list, removing three words per entry
;
Rem_Sel60:
        pop     dx                      ; get ENTRY flag ??
        or      dx,dx                   ; zero?
        jz      Rem_Sel70               ; if zero, this is end of list
        pop     dx                      ; remove this alternative
        pop     dx
        jmp     Rem_Sel60
Rem_Sel70:
        push    cs                      ; segment of parm list ptr
        push    si                      ; offset of parm list (buffer)
        push    ax                      ; selector for case
        push    cx                      ; put return segment back on
        push    bx                      ; and return offset
        retf                            ; and leave

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  This is called by the generated code to indicate end of an accept body.
;  When the rendezvous complete call is made, determine if the caller was
;  on my processor.  If not, use the Remote end accept, otherwise use the
;  local end accept
;               Inputs:
;                       No user inputs, only the REPLY pointer
;                       provides information regarding the responding task.
;
Rendezvous_Complete:
        push    ds
        mov     ds,[VRTIF_DS]           ; get runtime data segment
        mov     si,[VRTIF_TASK_PTR]
        mov     si,VRTIF_tcbtid[si]     ; fetch my TID
        pop     ds
        add     si,si                   ; mult by four to make index
        add     si,si
        test    cs:TASK_DIRECTORY.DTCB_dir_LOCAL[si],0FFFFH ; distributed?
        jnz     Dist_End_Accept         ; must to a distributed accept
        jmp     VRTIF_Rendezvous_Complete ; otherwise, return to vendor runtime


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
;    Distributed_End_Accept -
;         Reply ptr has got the buffer descriptor, use it to determine
;         if call was local or remote
;
DEA_ENTRY        equ     -2       ; local word for entry pointer

Dist_End_Accept:
;         mov     al,'e'
;         call    Outchr          ; @@
        push    bp
        mov     bp,sp
        sub     sp,2                    ; local data
        push    ds                      ; save previous DS
        push    cs                      ; load data segment
        pop     ds
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[si] ; fetch TCB of my task
        mov     di,DEF_tcb_reply[si]            ; fetch buffer descriptor
        mov     di,[di]

        mov     ax,DEF_pkt_entry_id[di]         ; fetch Entry id
        mov     bx,ax
        add     bx,bx                           ; mult by 2
        add     bx,bx                           ;  * 4
        add     bx,bx                           ;  * 8
        lea     bx,DEF_entry_table[si+bx]       ; point to entry
        mov     [bp+DEA_ENTRY],bx               ; save entry record ptr
        lea     bx,DEF_entry_queue[bx]          ; point to entry queue
        call    REMOVE                  ; pull entry off queue BX now @ buffer

        cmp     DEF_pkt_cmd[di],DEF_local_call  ; see if this is local
        jz      Local_End_Accept

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    Send output parameters to caller.
;    Release buffer used to hold input (and output for now) parameters.
;
;       INPUT:  SI is my TCB address
;               DI points to buffer used for this rendezvous
;               BX points to buffer descriptor
;
;       NOTE: Stack frame is already build for local parameters
;
Remote_End_Accept:
;         mov     al,'R'
;         call    Outchr
        push    bx                      ; save buffer descriptor
;
; Build stack for XMIT
;
; PARM LIST PTR
```

```
        push    cs                          ; segment of buffer
        push    di                          ; first part of buffer is parm list
; MODE
        mov     ax,DEF_out                  ; out mode
        push    ax
; PROFILE
        mov     bx,[bp+DEA_ENTRY]           ; get base of this entry
        mov     bx,DEF_entry_profile_ptr[bx]     ; fetch prorile
        push    bx
; MY_TID
        mov     ax,[di+DEF_pkt_TID]         ; This task was the orignial TID
        push    ax
; ENTRY
        mov     ax,[di+DEF_pkt_entry_ID]
        push    ax
; TID
        mov     di,[di+DEF_pkt_MY_TID]  ; get caller's task ID
        push    di
; CMD
        mov     ax,DEF_rendezvous_end
        push    ax
; PID
        add     di,di                       ; mult TID by four to make index
        add     di,di
        mov     ax,TASK_DIRECTORY.DTCB_dir_pid[di]      ; fetch PID
        push    ax
        call    IO_Xmit                     ; transmit reply
;
;  Now we are done with the received buffer, release it
;
        pop     bx                          ; get descriptor ptr back
        call    IO_Deallocate               ; release buffer, descriptor in BX
        pop     ds                          ; restore DS
        mov     sp,bp
        pop     bp
        retf


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Local_End_Accept
;   Allow caller to continue (Note: this is for entry calls with parameters
;   that are all passed by reference. No copy-back is required).
;   All entries whether remote or local use a buffer, therefore deallocate
;   it when complete.
;      INPUT:  SI is this task's TCB address
;              DI points to buffer used for this rendezvous
;              BX points to buffer descriptor
;
;      NOTE: Stack frame is already build for local parameters
;
```

```
Local_End_Accept:
;
;  Now wake up caller
;
;        mov     al,'L'
;        call    Outchr
        mov     si,DEF_pkt_my_tid[di]   ; get TID of caller
        call    IO_Deallocate           ; done with buffer deallocate @ BX
        add     si,si                   ; mult by four to make index
        add     si,si
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[si]      ; get TCB of caller
        push    cs                      ; push segment of semaphore
        push    si                      ; push calling Task's TCB (SEMAPHORE)
        call    VRTIF_Signal            ; signal task to continue
        pop     ds                      ; restore DS
        mov     sp,bp
        pop     bp
        retf


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Net_Receive - processes an incoming message           ;
;                                                         ;
;  This routine is called by the interrupt handler (in   ;
;  the IO Module) to initiate action based on the        ;
;  receipt of a packet.  When the service handler is     ;
;  called, BX contains the address of the buffer         ;
;  descriptor.                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Net_Receive:
        mov     si,[bx]                 ; get address of actual buffer
        mov     di,[si+DEF_pkt_cmd]     ; fetch command
        or      di,di                   ; do range check
        js      Net_Receive_Error
        cmp     di,command_limit
        jg      Net_Receive_Error
        shl     di,1                    ; make command into word index
        jmp     vector[di]
Net_Receive_Error:
        mov     al,'$'
        call    Outchr
        call    IO_deallocate           ; trash message
        ret
;
;  The following vector table implements the 'case' statement
;  on the message ACTION Field
;
vector  label   word
        dw      offset  Sync_Start
        dw      offset  Sync_Ready
```

```
        dw      offset  Sync_Continue
        dw      offset  Entry_Call
        dw      offset  Rendezvous_End
        dw      offset  Shut_Down               ; COLD START
vector_end      label   word
command_limit   equ     (vector_end - vector) / 2 - 1
;
;  Future versions of the vector table will include
;         Begin_Remove_Entry
;         End_Remove_Entry
;         Begin_Abort
;         End_Abort
;         Begin_Terminate
;         End_Terminate
;         Shared_Variable_Request
;        etc.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  This code section is executed upon receipt of a message initiating
;  a Begin_Elaborate request.  BX points to buffer descriptor.
;  NOTE: THIS IS ONLY RECIEVED BY SLAVES!
Sync_Start:
        call    IO_Deallocate           ; no need for buffer
        push    cs                      ; wait up slave
        lea     ax,SYNCHRO_SEMAPHORE
        push    ax
        call    VRTIF_Signal_I          ; signal task to continue
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  This code section is executed upon receipt of a message initiating
;  an End_Elaborate.  This message implies that the specified elaboration has
;  been completed on the remote processor and elaboration can continue
;  on the primary processor.
;
;  INPUTS: BX points to buffer descriptor.
;  NOTE: THIS IS ONLY RECEIVED BY THE MASTER
Sync_Ready:
;
        call    IO_Deallocate           ; no need for buffer
        push    cs                      ; wait up slave
        lea     ax,SYNCHRO_SEMAPHORE
        push    ax
        call    VRTIF_Signal_I          ; signal task to continue
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
; Sync_Continue: Executed when a "sync_continue" message arrives..
;       NOTE: ONLY RECEIVED BY SLAVES, half way through syncrhonization
;
Sync_Continue:
        call    IO_Deallocate           ; no need for buffer
        push    cs                      ; wait up slave
        lea     ax,CONTINUE_SEMAPHORE
        push    ax
        call    VRTIF_Signal_I          ; signal task to continue
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  This code section is executed upon receipt of a message initiating
;  an entry call
;
;  Place buffer on Entry queue, If "Waiting" for that entry is TRUE,
;  then clear all Waiting Flags and signal Wait Semaphore.
;   INPUTS:
;       BX = Buffer descriptor pointer
;       SI = Buffer pointer
;
;
;  This code assumes only a single parameter (simplification for prototype)
;               NOTE: pointers to data and descriptors are stored backward
;                     from normal Intel OFFSET,SEGMENT format
;
type_len        equ     4               ; offset to type/len field in profile
data_seg        equ     0               ; position within buffer for ptr to data
data_off        equ     2
desc_seg        equ     4               ; position within buffer for ptr to desc
desc_off        equ     6

true_data       equ     6               ; offset for data (after descriptor)

Entry_Call:
        mov     dx,bx                   ; save buffer descriptor
        mov     bx,[si+DEF_pkt_tid]     ; get task id
        add     bx,bx                   ; mult by four to make index
        add     bx,bx
        mov     bx,TASK_DIRECTORY.DTCB_dir_TCB[bx] ; get task control block
        mov     ax,[si+DEF_pkt_Entry_ID]; fetch entry id
        mov     di,ax                   ; compute entry offset
        add     di,di                   ; mult times 2
        add     di,di                   ; times 4
        add     di,di                   ; times 8
        lea     di,DEF_entry_table[di+bx]  ; point to current entry
        push    di
;
;   currently only one parameter is used (either in or out).  Take advantage
```

```
;   of this to simplify interface to accepting task.  The address of the
;   data area is provided in the first part of the buffer.  NOTE: this address
;   is backwards (segment=low address, offset=high address).
;
        mov     di,DEF_entry_Profile_Ptr[di] ; point to parameter profile
        test    [di+type_len],0FFFFH    ; see if constrained
        pop     di                      ; restore entry pointer
        jns     Entry_010               ; go on if constrained
;
;  Parameter is unconstrained, first pointer is to data, second to descriptor
;  The data will actually be offset by six (6) bytes to leave room for a
;  descriptor in front of the packet data.
;
        mov     data_seg[si],cs         ; stuff cs of buffer
        lea     ax,DEF_pkt_data+true_data[si]   ; address of true data
        mov     data_off[si],ax         ; put in packet
        mov     desc_seg[si],cs               ; segment of descriptor
        lea     ax,DEF_pkt_data[si]     ; offset of descriptor
        mov     desc_off[si],ax
        jmp     Entry_020
;
;  Handle simple case of constrained array
;
Entry_010:
        mov     data_seg[si],cs         ; stuff cs of buffer
        lea     ax,DEF_pkt_data[si]     ; address of data
        mov     data_off[si],ax         ;
;
;  ATOMIC action follows...  Queue entry, if waiting signal acceptor
;
Entry_020:
        xchg    bx,dx                   ; bx := buffer; dx := TCB_base
        lea     si,DEF_entry_queue[di]  ; si points to entry queue
        pushf
        cli
        call    INSERT                  ; place buffer descriptor on entry Q
        mov     cx,DEF_entry_wait[di]   ; get entry WAITING flag
        or      cx,cx                   ; test waiting flag
        jz      Entry_040               ; go on if not
;
;  server is waiting on accept, clear all waiting flags and signal it
;
        mov     si,dx                   ; get TCB
        mov     cx,DEF_num_entries[si]
Entry_030:
        mov     DEF_entry_table+DEF_entry_wait[si],0 ; clear wait flag
        add     si,size DEF_entry_rec   ; go to next entry
        loop    Entry_030

        push    cs                      ; segment of semaphore
        push    dx                      ; offset of semaphore (first in TCB)
```

```
        call    VRTIF_Signal_I          ; wake up server
;
Entry_040:
        popf                            ; restore interrupt level
        ret                             ; return to interrupt handler

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Rendezvous_End -
;  This code section is executed upon receipt of a message completing
;  an accept body (end rendezvous)
;
;  Post buffer containing Out Parameters and signal task to wake up
;
;   INPUTS:
;      BX = Buffer descriptor pointer
;      SI = Buffer Pointer
;
;
Rendezvous_End:

        mov     si,[si+DEF_pkt_tid]     ; fetch task id of caller
        add     si,si                   ; mult by four to make index
        add     si,si
        mov     si,TASK_DIRECTORY.DTCB_dir_TCB[si] ; fetch task control block
        mov     [si+DEF_TCB_REPLY],bx   ; provide caller with reply buffer
        push    cs                      ; push segment of caller semaphore
        push    si                      ; push offset of  same (TCB)
        call    VRTIF_Signal_I          ; wake up caller
        ret                             ; to finish interrupt


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                       ;
;  REMOVE - Remove Entry that is on entry queue                         ;
;                                                                       ;
;  Inputs:  BX points to entry Q                                        ;
;  Output:  BX points to buffer descriptor that was dequeued            ;
;                                                                       ;
;  All other registers are preserved                                    ;
;                                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
REMOVE:
        push    ax
        push    si
;
;  do list operation as atomic action
;
        pushf
        cli
        mov     si,[BX+DEF_NEXT_PTR]    ; fetch buffer descriptor
        mov     ax,[si+DEF_NEXT_PTR]    ; get next buffer
```

```
        mov     [BX+DEF_NEXT_PTR],ax    ; update queue head
        popf
        mov     bx,si                   ; return pointer in BX
        pop     si
        pop     ax
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                       ;
;  INSERT - INSERT Entry onto the end of an entry queue                 ;
;                                                                       ;
;  Inputs:  SI points to entry Q                                        ;
;           BX points to buffer descriptor                              ;
;                                                                       ;
;  Outputs: SI points to last entry on Q                                ;
;                                                                       ;
;  All other registers are preserved                                    ;
;                                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INSERT:
        push    ax
;
;  do list operation as atomic action
;
        pushf
        cli
INSERT10:
        mov     ax,[si+DEF_next_ptr]    ; get next buffer on entry queue
        or      ax,ax                   ; see if end of list
        jz      INSERT20                ; end of list, go insert it
;
;  this is not end of list, keep searching
;
        mov     si,ax
        jmp     INSERT10
;
;  found spot on list, insert it
;
INSERT20:
        mov     [si+DEF_next_ptr],bx    ; put on end of list
        popf                            ; restore interrupt flag
        pop     ax
        ret

        align   4

DUMMY_SEM   dw      3 dup (?) ; dummy semaphore for making zombie tasks

cseg    ends

        end
```

THIS PAGE INTENTIONALLY LEFT BLANK.

```
     page    55,132
     TITLE   IO - Distributed Ada Network IO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA_IO.ASM                                                           ;
;  IO  MODULE - Low Level Network Functions                                   ;
;                                                                             ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT   USA                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                             ;
;  The IO module provides the low_level interface to the network              ;
;  hardware and receive message buffering.                                    ;
;                                                                             ;
;  This code is loaded into all processors, and adapts to the                 ;
;  the network hardware in its host.  Which routines are used is              ;
;  determined solely by the calls made from the application code              ;
;  and the messages received.                                                 ;
;                                                                             ;
;  The IO interface is implemented as four separate functions:                ;
;     Initialize                                                              ;
;     Transmit                                                                ;
;     Receive                                                                 ;
;     Interrupt Procesing                                                     ;
;                                                                             ;
;  The initialize function obviously must be called prior to any              ;
;  other, and establishes the interrupt vector and enables, as                ;
;  well as prepares the hardware for use.  It is also responsible             ;
;  for initilizing data structures used to buffer incoming packets.           ;
;                                                                             ;
;  The Transmit function is used by one task at a time, and is                ;
;  guarded by a semaphore to provide mutual exclusion.  Once the              ;
;  transmit resource is granted, the data is copied into the on-card;
;  buffer and sent out via hardware commands.  (Normally, hardware            ;
;  packet acknowledge should be provided, however Ethernet does not           ;
;  support this, so we have implemented an acknowledge with time-out;
;  protocol that provides network error detection.  Note that                 ;
;  acknowledement packets take priority over regular traffic.                 ;
;  Currently, no re-try is supported, however it would be a rather            ;
;  simple matter of keeping a transmit buffer queue and retransmit            ;
;  on errors.  The more serious problem is how to insure real-time            ;
;  performance in the presence of multipie retries.  Obviously the            ;
;  retry count would have to be programmable (and possibly time               ;
;  sensitive.  If an acknowledgment times-out, a reconfiguration              ;
;  operation is executed to recover the system in a reduced state.            ;
;                                                                             ;
;  The Receive function is provided to assist in transferring the             ;
;  data to the requested destination.  It clears the outstanding              ;
;  acknowledgement request.                                                   ;
;                                                                             ;
;  The Interrupt Processing handles both transmit complete and                ;
;  reception interrupts.  For transmit complete, the resource is              ;
;  simply made available again by performing a V operation on the             ;
```

```
; trasmit semaphore.  For Receive interrupts, a buffer is allocated;
; from a linked list of fixed sized buffers.  Then the incoming    ;
; data is copied to the buffer and the distributed runtime is      ;
; invoked to process the request.  It may simply post the fact the ;
; message has arrived (and queue to an entry), or it may cause a   ;
; task to resume which involves signalling (V - operation) the     ;
; suspended task.                                                  ;
;                                                                  ;
; Refer to individual procedure headers for parameter information  ;
; and calling requirements.                                        ;
;                                                                  ;
;                                                                  ;
; Ver Date      Description                                        ;
;                                                                  ;
; 0.1 Nov-88 : Initial prototype                                   ;
; 0.2 Dec-89 : Added Packet Acknowledge/Error Detection, and       ;
;              allowed for system restart (compiler initialized    ;
;              data is restricted to a WARM_START flag.)           ;
; 0.3 Feb-90 : Greatly improved multi-packet processing and        ;
;              interrupt handling.                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .model  large
        include DA_DEF.ASM              ; contains software definitions
        include DA_HW.ASM               ; contains hardware specifics

        public  IO_Network_Init, IO_Xmit
        public  TX_READY                ; semaphore
        public  IO_ALLOCATE, IO_DEALLOCATE
        public  Ack_Check
        public  outchr                  ; for debug
        public  RECEIVE_FLAG            ; for sync_phase IO

        extrn   VRTIF_Signal_I:far      ; signal semaphore
        extrn   VRTIF_Wait:far          ; wait on semaphore "P"
        extrn   VRTIF_I8259:abs         ; address of 8259
        extrn   VRTIF_vector_base:abs   ; base of vector table
        extrn   VRTIF_timestamp:far     ; time stamping routine
        extrn   Setup:near              ; Initialize Network I/F
        extrn   NET_Receive:near        ; part of runtime code
        extrn   Shut_Down:near          ; if network ack failure
        extrn   COLD_START:word         ; NZ if this is first pass thru

        extrn   SYNC_PHASE:word         ; determines operational phase
        extrn   NET_TABLE:byte          ; provides network addresses
        extrn   PID:word                ; THIS processor's ID #

        extrn   TASK_DIRECTORY:word     ; (DTCB) table of tasks
        extrn   WATCH_DOG:word          ; (DTCB) table of watch dog timers
        extrn   WATCH_LIST:word         ; (DTCB) list of processors to watch
```

```
;
;  software support buffers
;
buff_size              equ    2048    ; bytes in local buffer
num_buff               equ    20      ; number of buffers
min_packet             equ    64      ; minimum number of bytes in a packet

cseg    segment common

        org     2800H                 ; makes listings eaiser to use!
        assume  cs:cseg,ds:cseg,es:cseg ; ,ss:sseg

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   NETWORK_INIT  : load Interrupt Vector and clear pointers  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IO_Network_Init:
        push    ax
        push    bx
        push    cx
        push    dx
        push    ds

;
;  Do low level Network Interface Card Initialization
;
        call    Setup
;
;  init network variables
;
        mov     [SEQUENCE],0                    ; zero out sequnce counter
        mov     ax,cs
        mov     ds,ax
        mov     [RECEIVE_PTR],eth_recv_begin   ; receive pointer

        mov     [TX_READY],1                   ; init semaphore
        mov     [TX_READY+2],0
        mov     [TX_READY+4],0
        mov     [RECEIVE_FLAG],0                ; init flag
;
;  Initialize Receive buffer list
;
        lea     ax,RX_BUFF_Q
        mov     [RX_BUFF_HEAD],ax

        lea     ax,RX_BUFFER    ; points to actual buffers
        mov     cx,num_buff     ; number to link
        lea     bx,RX_BUFF_Q    ; points to buffer descriptors
Init30:
        mov     [bx],ax         ; put in current buffer pointer
        lea     dx,[bx+4]       ; DX is address of next descriptor
        mov     [bx+2],dx       ; put it in as next pointer
```

```
        add     ax,buff_size     ; point AX at next buffer
        mov     bx,dx            : change descriptor pointer to next
        loop    Init30
;
;  now fix up last pointer
;
        mov     word ptr [bx-2],0        ; terminate list


;
;  Initialize Outstanding Acknowledgements lists
;
        xor     ax,ax                    ; indicate none outstanding
        mov     [ACK_PENDING],ax
        mov     [ACK_HOLDING],ax         ; or waiting to be xmitted

        lea     si,ACK_RECORDS           ; all acks are free
        mov     [ACK_FREE],si
        mov     cx,num_buff-1    ; number to link (same as number of buffers)
Init40:
        lea     dx,[si+ack_size]; DX is address of next descriptor
        mov     [si],dx          ; put it in as next pointer
        mov     si,dx            ; change descriptor pointer to next
        loop    Init40
;  fix up last pointer
        mov     word ptr [si],0         ; terminate list


;
;  load interrupt vector if this is a cold start
;
        test    word ptr [COLD_START],0FFFFH
        jz      Warm_Start
        mov     ax,0
        mov     ds,ax
        mov     bx,VRTIF_vector_base+(vector_number*4)
        mov     ax,offset Interrupt_Handler
        mov     [bx],ax
        mov     ax,cs
        mov     [bx+2],ax
;
;  Note: Preliminary board initialization was done in SETUP code, now
;        just enable interrupts
;
Warm_Start:
        mov     dx,VRTIF_I8259+1
        in      al,dx                    ; get interrupt mask
        mov     ah,0FEH                  ; mask to clear zero bit
        mov     cl,vector_number         ; load shift count register
        rol     ah,cl
        and     al,ah                    ; enable level
        out     dx,al                    ; update controller chip
```

```
        mov     dx,nic_cr               ; command register
        mov     al,eth_access_page_0    ; access NIC page 0 registers
        out     dx,al

        mov     dx,nic_imr              ; interrupt mask register
        mov     al,nic_prx+nic_ptx      ; enable xmit/recv interrupts
        out     dx,al

        pop     ds
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret

;
;  routine for debugging only - all registers preserved
;  Prints character in AL
;
outchr:
        push    dx
        push    ax
        mov     dx,3fdh
out10:
        in      al,dx
        and     al,20h
        jz      out10
        pop     ax
        mov     dx,3f8h
        out     dx,al
        pop     dx
        ret


header_size     equu    10 ;words:dst=3,src=3,RCP=1,priority=1,seq=1,length=1

rcp_offset      equ     12 ; bytes to receive control pointer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  XMIT -  transmit the message specified by parameter list        ;
;          starting at address is at SS:bp+DEF_PARM_LIST           ;
;   NO GENERAL REGISTERS ARE PRESEVED                              ;
;  NOTE: During system synchonization this routine works           ;
;        differently so as to avoid use to the vendor runtime and ;
;        provide more control to the appplication (no ack timeout);
;        This is designated by the boolean  "SYNC_PHASE"           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  INPUTS:
;               PID             ; destination processor ID
;               CMD             ; command for this packet
;               TID             ; Task for which the command operates
```

```
;               ENTRY           ; entry ID for the command (if applicable)
;               MY_TID          ; originating Task ID
;               PROFILE         ; profile pointer (in CS) for entry parameters
;               MODE            ; current calling mode (in or out)
;               PARM_LIST       ; pointer (seg/offset) for parameter list
;
IO_Xmit:
        push    bp
        mov     bp,sp           ; mark stack
        test    [SYNC_PHASE],0FFFFH ; see if in sync phase, if so, don't use rts
        jnz     Xmit_05
;
;   Normally, we use vendor runtime to lock xmitter
;
        push    cs                  ; push segment of transmit ctrl semaphore
        lea     ax,TX_READY
        push    ax                  ; push offset of semaphore
        call    VRTIF_Wait          ; do p semaphore operation
;
;  Now get acknowledge request buffer
;
        call    Ack_Allocate        ; returns ack buffer ptr in BX ;AAA
        jmp     Xmit_08
;
;   But During SYNC_PHASE we simply lock with a clear
Xmit_05:
        mov     [TX_READY],0          ; set it not busy, set by Interrupt rtn
Xmit_08:
;
;  put header in packet buffer
;
        cld                         ; set auto increment
        les     di,[CARD_RAM]       ; point to hardware buffer area
        mov     si,[bp+DEF_PID]     ; fetch Destination Task PID
        mov     cl,3                ; mult by 8 (bytes/address entry)
        shl     si,cl               ; index into address table
        lea     si,NET_TABLE[si]    ; fetch address of dest.
        mov     cx,DEF_addr_size    ; in words
        rep     movsw               ; copy in dest address
        mov     si,[PID]            ; get our processor id
        mov     cl,3                ; mult by 8 (bytes/address entry)
        shl     si,cl               ; index into address table
        lea     si,NET_TABLE[si]    ; fetch our address
        mov     cx,DEF_addr_size
        rep     movsw               ; copy in source addr
;
;  skip over length field for now
;
        add     di,2
;
;  Update Sequence Number and put it in packet and acknowledge entry
```

```
;
        mov     ax,[SEQUENCE]       ; get sequence number
        inc     ax
        stosw                       ; put in packet
        mov     [SEQUENCE],ax       ; update
        test    [SYNC_PHASE],0FFFFH ; see if in sync phase
        jnz     Xmit_09
        mov     [bx+ACK_SEQ],ax     ; put it in outstanding requests ;AAA
        call    Ack_Add             ; add this ack entry to the pending list ;AAA
Xmit_09:
        mov     ax,[bp+DEF_CMD]     ; get packet command
        stosw                       ; put in buffer
        mov     ax,[bp+DEF_TID]     ; get Destination TID
        stosw
;
        mov     ax,[bp+DEF_ENTRY]      ; if entry applies
        stosw
        mov     ax,[PID]                ; fetch my processor ID
        stosw
        mov     ax,[bp+DEF_MY_TID]     ; get my task ID
        stosw
;
;  copy the parameters into the packet buffer.  Use the TCB definitions
;  to determine how many parameters, their size, and what type (ie. must
;  allow for unconstrained arrays).
;
        mov     si,[bp+DEF_Profile]     ; get parameter profile ptr
        lodsw                           ; get number of parameters
Xmit_10:
        or      ax,ax                   ; see if done
        jz      Xmit_30                 ; if done
        mov     [XPARM_COUNT],ax        ; update parameter count
        mov     ax,[si]                 ; get parameter Mode
        mov     cx,[si+2]               ; fetch parameter type/length
        add     si,4
        mov     [PROFILE_PTR],si        ; save profile pointer

        lds     si,[bp+DEF_parm_list]   ; point to parameter list
        push    [si]                    ; segment of data
        push    [si+2]                  ; offset of data

        or      cx,cx                   ; see if unconstrained type
        jge     Xmit_15
;
;  process an unconstrained object as a parameter, always copy descriptor
;
        push    [si+4]                  ; descriptor segment
        push    [si+6]                  ; offset of descriptor
        add     si,8
        mov     word ptr [bp+DEF_parm_list],si ; update parameter list index
        pop     si                      ; get offset of descriptor
```

```
        pop     ds                      ; get segment of descriptor
        push    ax                      ; save MODE of parameter
        mov     cx,[si+DEF_low_desc]    ; get low bound of constraint
        mov     es:[di],cx              ; put in packet
        add     di,2
        mov     ax,[si+DEF_high_desc]   ; get high bound of constraint
        stosw                           ; put in packet
        mov     dx,[si+DEF_size_desc]   ; get size of object
        mov     es:[di],dx              ; put in packet
        add     di,2
;
; Copy the parameter data iff MODE is correct and array is not null
;
        pop     bx                      ; get mode of parameter
        pop     si                      ; get offset of data
        pop     ds                      ; get segment of data
        and     bx,[bp+DEF_MODE]        ; see if we should copy data
        jz      Xmit_20                 ; if not, go on
        sub     ax,cx                   ; compute difference in range
        inc     ax                      ; adjust to include end points
        jle     Xmit_20                 ; if array is empty go to next parm
        mul     dx                      ; compute size in words (descriptor)
        mov     cx,ax                   ; put in count register
        rep     movsw                   ; transfer to packet buffer
        jmp     Xmit_20                 ; go on to next parameter
;
;  Constrained parameter, CX is length in words, copy it into packet buffer
;
Xmit_15:
        add     si,4                    ; move to next object address
        mov     word ptr [bp+DEF_parm_list],si ; update parameter list index
        pop     si                      ; get data offset
        pop     ds                      ; get data segment
        and     ax,[bp+DEF_MODE]        ; see if mode is right
        jz      Xmit_20                 ; skip copy of data if not
        inc     cx                      ; round up to nearest word count
        shr     cx,1                    ; by adding one and divide by two
        rep     movsw

Xmit_20:
        mov     ax,cs                   ; restore data segment
        mov     ds,ax
        mov     si,[PROFILE_PTR]        ; get next parameter profile
        mov     ax,[XPARM_COUNT]        ; get the counter back in ax
        dec     ax                      ; count down
        jmp     Xmit_10

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Setup NIC registers to begin transmission                              ;
; Must prevent a RECEIVE interrupt from arriving, which would interfere  ;
; with the registers being updated for Transmission.                     ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  load start address of packet
;
Xmit_30:
        pushf                           ; save interrupt status
        cli                             ; disable any interrupts
        mov     dx,nic_cr               ; select Page_0
        mov     al,eth_access_Page_0
        out     dx,al

        mov     dx,nic_tpsr             ; page start register
        mov     al,eth_xmit_buf_start   ; transmit page at DC00:0000
        out     dx,al
;  load length of packet
        mov     ax,di                   ; save current packet pointer
        les     di,[CARD_RAM]           ; point to hardware buffer area
        sub     ax,di                   ; subtract base to get size in bytes
        add     di,DEF_pkt_length       ; add offset to data length field
        stosw                           ; stick in PACKET length
        cmp     ax,min_packet           ; make sure it is at least minimum
        jge     Xmit_40
        mov     ax,min_packet
Xmit_40:
        mov     dx,nic_tbcr0            ; load number to transfer into H/W
        out     dx,al
        mov     dx,nic_tbcr1
        mov     al,ah
        out     dx,al
;  start transmit
        mov     dx,nic_cr
        mov     al,send         ; command to initiate transmission
        out     dx,al
        popf                    ; restore interrupt status
        pop     bp              ; restore bp
        ret     18              ; return and remove stack frame

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   INTERRUPT SERVICE ROUTINE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Currently, this must have a stack frame similar to other vendor
;  interrupt routines so that the interrupt-mode Signal routine will
;  be able to find the interrupt return address and status
;
Interrupt_Handler       label   far
        push    bp
        mov     bp,sp
        push    ax
```

```
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es
;
;  First keep interrupt request line from triggering during processing
;  of interrupts (and clearing interrupt bits)
;
        cld                     ; for all string operations
        mov     dx,nic_cr       ; select Page_0
        mov     al,eth_access_Page_0
        out     dx,al

        mov     dx,nic_imr              ; interrupt mask register
        mov     al,eth_ints_disabled    ; disable all interrupt requests
        out     dx,al
;
;  Process any packet receptions
;
;  NOTE: since this is done inside the interrupt routine, interrupts
;  are disabled, and therefore there is no interference from other
;  interrupts is expected (especially clock interrupts).
;  Careful attention to race conditions is necessary to prevent a received
;  buffer from not getting processed and interrupts getting lost
;
Receive:
;   point DS:SI to packet in hardware buffer

        lds     si,cs:[CARD_RAM]   ; source is ethernet RAM
        add     si,cs:[RECEIVE_PTR]; add current receive buffer page address

        mov     ax,[si]         ; fetch status into AL, NEXT PTR into AH
                                ; since we only receive good packets ignore stat
        or      al,al           ; see if any packet arrived (if not zero!)
        jnz     RECV100         ; go on if data is there
        jmp     End_Receive     ; otherwise, leave the receive section
;$$$;
;$$$;  No data left, go ahead and clear receive interrupt
;$$$;      RACE CONDITION HERE...
;$$$        mov     dx,nic_isr      ; clear any pending receive interrupts
;$$$        mov     al,nic_prx      ; receive interrupt bit
;$$$        out     dx,al           ; clear receive interrupt (if present)
;$$$;
;$$$;   put in little delay, then make sure nothing just arrived..
;$$$;
;$$$        mov     ax,10
;$$$RECV020:
;$$$        dec     ax
```

```
;$$$        jnz     RECV020
;$$$        mov     al,[si]         ; see if something has arrived
;$$$        or      al,al
;$$$        jz      RECV030         ; if nothing, good... no worries
;$$$;  something did just arrive, see if we will see the interrupt
;$$$        in      al,dx           ; fetch interrupt status now
;$$$        and     al,nic_prx      ; see if a receive interrupt was shut off
;$$$        jz      RECV040         ; if we lost the interrupt go sound alarm
;$$$RECV030:
;$$$        jmp     Check_Xmit
;$$$;
;$$$;  We shut off a receive interrupt by accident
;$$$;
;$$$RECV040:
;$$$        mov     al,7            ; @@@ print bell
;$$$        call    outchr
;$$$        mov     al,'_'          ; visible evidence
;$$$        jmp     Check_Xmit
;$$$
RECV100:                        ; @@@ check for non recieve ok ptr
        cmp     al,1            ; is it a one?
        jz      RECV101
        mov     al,7            ; @@@ print bell
        call    outchr
        mov     al,'?'          ; if non-zero print something special
        call     outchr
RECV101:
        xor     al,al           ; zero low byte, leaving a new pointer
        sub     ax,eth_offset   ; correct for memory vs page offset
        mov     cs:[RECEIVE_PTR],ax ; get ready for next reception
        add     si,4            ; skip over receive header (status/page, count)
;
;  SI now points to first part of transmitted packet
;
;  First check to see if it is an ACK message
;
        mov     ax,[si+DEF_pkt_cmd] ; check message type
        or      ax,ax           ; command is negative for Acks
        jns     RECV105         ; if regular packet, go on
;
;   It is an ACK message.  Clear it from pending list and free up buffer
;
        mov     byte ptr [si-4],0  ; clear status flag for next time
        call    ACK_REMOVE      ; check off the ack
        jmp     END_RECEIVE     ; all done with one packet

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Received a real message, first reload watchdog timer for the source PID
RECV105:
        mov     di,[si+DEF_pkt_my_pid]   ; get source processor ID
        add     di,di
```

```
        mov     cs:WATCH_DOG[di],DEF_WATCH_DOG_LIMIT
;
; Allocate a buffer, and transfer data to the buffer
; after the following call, the buffer descriptor is in BX. DO NOT DESTROY BX!
;
        mov     ax,cs           ; destination segment is CS
        mov     es,ax
        call    IO_Allocate     ; destination offset is buffer header in BX
        mov     di,cs:[bx]      ; get address of buffer in DI
        mov     ax,[si+DEF_pkt_length] ; get size of valid packet in bytes
        inc     ax              ; make sure we get odd packets
        shr     ax,1            ; convert to words
;
; Now transfer memory from hardware buffer pages to software buffer.
; Note that the buffer will wrap around at 4000H back to 2600.
; Also, the first word of each page is cleared after the data is removed
; so that received packets can easily be detected.  (Since the header bytes
; are the last thing written, you are guaranteed that the whole packet has
; been received.)
;
        mov     dx,80H-2        ; page size in words (reduced to get aligned)
RECV110:
        cmp     ax,dx           ; see if more than a page
        jge     RECV120
        mov     dx,ax           ; otherwize only move the remaining words
RECV120:
        mov     cx,dx
        rep     movsw           ; do the transfer
        push    si
        dec     si              ; make sure we are in page just processed
        and     si,0FF00H       ; backup to its beginning
        mov     byte ptr [si],0 ; and clear status byte for next time
        pop     si

        cmp     si,eth_recv_end ; see if at end of hardware buffer
        jnz     RECV130
        mov     si,eth_recv_begin; reset pointer to begin
RECV130:
        sub     ax,dx           ; reduce total count by those moved
        jz      RECV140         ; finished if so
        mov     dx,80H          ; keep page alignment
        jmp     RECV110
RECV140:
        mov     ax,cs           ; restore data segment
        mov     ds,ax
;
; Check what phase we are in.  If Sync_Phase, do not ack the message or
; invoke the distributed runtime.
;
        test    [SYNC_PHASE],0ffffH     ; NZ means true (sync phase)
        jz      RECV150
```

```
;
; System is still in synchronization phase, simply log that the message
; arrived by setting the RECEIVE FLAG with the buffer descriptor
; During sync phase, only one message can be recieved, so no concern
; for overwriting the RECEIVE_FLAG exists.

        mov     [RECEIVE_FLAG],bx
        jmp     End_Receive             ; done receiving

; Queue an ACK for the sendor then...
; Call Receive portion of Distributed Runtime code to determine
; what should be done with the newly arrived packet.
;
RECV150:
        mov     si,[bx]         ; get beginning of buffer back
        call    Ack_Hold        ; first queue an Ack message to go out ;AAA
        call    NET_Receive
;
; END RECEIVE: do check on buffer, if no packet there, clear interrupts
;
End_Receive:
        lds     si,cs:[CARD_RAM]   ; source is ethernet RAM
        add     si,cs:[RECEIVE_PTR]; add current receive buffer page address
        mov     ax,[si]         ; fetch status into AL, NEXT PTR into AH
                                ; since we only receive good packets ignore stat
        or      al,al           ; see if any packet arrived (if not zero!)
        jz      Clear_Interrupt ; go on if no data is there
        jmp     Check_Xmit
;
; No data left, go ahead and clear receive interrupt
;     RACE CONDITION HERE...
Clear_Interrupt:
        mov     dx,nic_isr      ; clear any pending receive interrupts
        mov     al,nic_prx      ; receive interrupt bit
        out     dx,al           ; clear receive interrupt (if present)
;
;  Check if we won the race...
;
        mov     ax,10
CI10:
        dec     ax
        jnz     CI10
        test    byte ptr [si],0FFH      ; see if something just arrived
        jz      Check_Xmit
;
;  Something just arrived, see if we can see the interrupt
;
        in      al,dx           ; get interrupt status
        and     al,nic_prx
        jnz     Check_Xmit      ; ok, we still see the interrupt
        mov     al,7            ; print bell!
```

```
        call    outchr          ; interrupt has been lost! due to race
        mov     al,'x'
        call    outchr
;
;  Now check for transmit complete interrupt
;
Check_Xmit:
        mov     ax,cs           ; insure data segment is for DRT
        mov     ds,ax
        mov     dx,nic_isr      ; get interrupt status
        in      al,dx
        and     ax,nic_ptx      ; check for packet transmitted
        jnz     Transmit
;
;  No xmit complete interrupts, see if there is a ACK to go out
;
        test    [ACK_HOLDING],OFFFFH    ; see if any acks are waiting to go out
        jz      EOI             ; nothing to go out
        mov     ax,[TX_READY]   ; check if transmitter is busy
        or      ax,ax
        jle     EOI             ; still busy, just exit
        call    ACK_Send        ; otherwise send out one of the holding acks
        jmp     EOI
;
;  Transmit complete, see if an ACK is waiting to go out.  If so,
;  send it.  Otherwise signal READY semaphore.
;
Transmit:
        out     dx,al                   ; clear the transmit interrupt

        test    [ACK_HOLDING],OFFFFH    ; see if any acks are waiting to go out
        jz      transmit10
        inc     word ptr [TX_READY]     ; give credit for transmit complete
        call    ACK_SEND                ; go issue the ack
        jmp     EOI
;
;  Only free up transmitter if no acks waiting to go
;  (and out of sync phase,  Note: Acks never occur during sync phase)

transmit10:
        test    [SYNC_PHASE],OFFFFH;    if sync phase, indicate free xmitter
        jz      transmit20
        mov     [TX_READY],1            ; by setting it ready
        jmp     EOI
transmit20:
        push    cs              ; segment of semaphore
        lea     ax,TX_READY     ; offset of semaphore
        push    ax
        call    VRTIF_Signal_I  ; signal ready for next IO

;
```

```
; Interrupt processing has been completed.  Any new interrupts that have
; come in since clearing the status bit will be recorded by the 8259
; when we enable the 3Com card interrupt mask.  This creates the edge
; trigger necessary for the 8259
;
EOI:
;
; Clear the 8259 Interrupt Request
;
        cli                     ; make absolutely sure we don't nest
        mov     dx,nic_imr      ; point to mask register
        mov     al,nic_ptx+nic_prx ; enable transmit (tx) and receive (rx) ints
        out     dx,al
        mov     al,NET_EOI      ; issue EOI to interrupt controller
        mov     dx,VRTIF_I8259
        out     dx,al

        pop     es              ; restore registers and flags (interrupt)
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     bp
        iret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; IO_ALLOCATE - Allocates next buffer from Avail list                   ;
;      Return BX pointing to buffer queue index.                        ;
;      By design, the buffer should queue should never be empty.        ;
; Destroys AX   , BX has new descriptor pointer                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IO_ALLOCATE:
        pushf
        cli
        mov     bx,CS:[RX_BUFF_HEAD]    ; fetch head pointer
        or      bx,bx                   ; see if empty
        jnz     IO_ALLOC10              ; go on if not
;
; Normally, might raise storage error here, but design prevents
; exceeding buffer capacity unless there is some code flaw.
;
        popf
        mov     al,'M'                  ; print message
        call    outchr
        mov     al,'T'                  ;    and
        call    outchr
        int     3                       ; trap
```

```
;
; Remove buffer descriptor from free list
;
IO_ALLOC10:
        mov     ax,CS:[bx+DEF_NEXT_PTR] ; fetch next pointer
        mov     CS:[RX_BUFF_HEAD],ax    ; pull buffer off list, replace head
        xor     ax,ax                   ; null next pointer in buffer
        mov     CS:[bx+DEF_NEXT_PTR],ax
        popf
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; IO_DEALLOCATE - Deallocates buffer into Avail list                          ;
;       Takes BX pointing to buffer descriptor.                                ;
;     By design, the buffer should queue should never be full.                 ;
; Destroys AX                                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IO_DEALLOCATE:
        pushf
        cli
        mov     ax,[RX_BUFF_HEAD]       ; get head of list
        mov     [bx+DEF_NEXT_PTR],ax    ; put behind this entry
        mov     [RX_BUFF_HEAD],bx       ; make this entry new head
        popf
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; ACK_ALLOCATE - Allocates next buffer from Free list                         ;
;     Return BX pointing to Ack entry.                                         ;
;     By design, the free list should never be empty.                          ;
; Destroys AX  , BX has new descriptor pointer                                 ;
;     Interrupts are disabled to maintain list consistency                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ACK_ALLOCATE:
        pushf
        cli
        mov     bx,[ACK_FREE]           ; fetch head pointer
        or      bx,bx                   ; see if empty
        jz      ACK_ALLOC10             ; if failure
;
; Remove buffer descriptor from free list
;
        mov     ax,[bx+ACK_NEXT]        ; fetch next pointer
        mov     [ACK_FREE],ax           ; pull buffer off list, replace head
        xor     ax,ax                   ; null next pointer in buffer
        mov     [bx+ACK_NEXT],ax
        popf                            ; restore interrupts
        ret

;
; Normally, might raise storage error here, but design prevents
```

```
;  exceeding buffer capacity unless there is some code flaw.
;
ACK_ALLOC10:
        popf
        mov     al,'M'                  ; print message
        call    outchr
        mov     al,'T'                  ;    and
        call    outchr
        int     3                       ; trap

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  ACK_DEALLOCATE - Deallocates buffer into Free list                  ;
;       Takes BX pointing to buffer descriptor.                         ;
;      By design, the ack list should never be full prior to call.      ;
;  Destroys AX                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ACK_DEALLOCATE:
        pushf
        cl
        mov     ax,[ACK_FREE]           ; get head of list
        mov     [bx+ACK_NEXT],ax        ; put behind this entry
        mov     [ACK_FREE],bx           ; make this entry new head
        popf
        ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   ACK_ADD  - Add another ack entry to the pending list
;              Input: BX is ack entry
;              ax is destroyed
Ack_Add:
        push    si
        push    di

        push    ax                      ;&&& save regs
        push    cx
        push    di
        and     ax,0fffH                ; only use 0-4095
        push    ax                      ; push sequence # &&&
        call    VRTIF_Timestamp         ; time-stamp it   &&&
        pop     di
        pop     cx
        pop     ax                      ; &&& restore regs

        pushf
        cli
        mov     ax,cs:[ACK_TIMER]
        add     ax,ack_delay            ; number of ticks
        mov     cs:[bx+ACK_COUNT],ax
;
;  Find the end of the ack list
```

```
;
        lea     si,ACK_PENDING          ; point to header
ack_add10:
        mov     di,cs:[si]              ; fetch next pointer
        or      di,di                   ; see if at end
        jz      ack_add20               ; jump if so
        mov     si,di                   ; go down the list
        jmp     ack_add10

ack_add20:
        mov     cs:[si],bx              ; put behind this entry
        popf
        pop     di
        pop     si
        ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   ACK_REMOVE - Remove ack entry from the pending list.
;     THIS IS ONLY CALLED DURING RECEIVE INTERRUPT ROUTINE
;              SI : points to ACKNOWLEDGE PACKET BUFFER IN HARDWARE
;                   relative to the DS segment which points to the
;                   hardware packet buffer (NOTE: ACK's are never
;                   unloaded from the hardware buffer).
;              ax,bx,si,cx destroyed
;
;   NOTE: ACK's have the SEQUENCE they are acking in the normal
;         sequence field.
;
Ack_Remove:
        mov     cx,[si+DEF_pkt_sequence]   ; get SEQUENCE value

        push    ax                      ;&&& save regs
        push    cx
        push    di
        and     cx,0fffH                    ; only use 0-4095
        push    cx                          ; &&& push for timestamp
        call    VRTIF_timestamp             ; &&&
        pop     di
        pop     cx
        pop     ax                      ; &&& restore regs

        lea     bx,ACK_PENDING
Ack_remove10:
        mov     si,cs:[bx]                  ; get next pointer
        or      si,si                       ; exit if at end of list
        jz      ack_remove30                ; all done (not there!!)
        cmp     cx,cs:[si+ack_seq]          ; check for matched sequence
        jz      ack_remove25
ack_remove20:
        mov     bx,si                       ; bx is always the previous pointer
```

```
        jmp     ack_remove10
;
;  Found the entry, remove from pending, and place it on FREE list
;
ack_remove25:
        mov     ax,cs:[si]              ; get next in list
        mov     cs:[bx],ax              ; link over removed entry
;  put removed node into free list
        mov     ax,cs:[ACK_FREE]        ; get head of list
        mov     cs:[si],ax              ; put behind this entry
        mov     cs:[ACK_FREE],si        ; make this entry new head
ack_remove30:
        ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   ACK_HOLD  - Add another ack message to the holding list
;               Input: SI points to received message
Ack_HOLD:
        push    bx                    ; save message descript--
        call    ACK_ALLOCATE          ; fetch a free ack entry
        mov     ax,[si+DEF_pkt_my_pid]  ; get pid of sender
        mov     [bx+ack_pid],ax       ; put in record
        mov     ax,[si+DEF_pkt_sequence]; fetch received Sequence #
        mov     [bx+ack_seq],ax       ; stick in record
;
;   put at end of HOLDING list
;
        push    si
        push    di
        lea     si,ACK_HOLDING
ack_hold10:
        mov     di,[si]               ; fetch ptr
        or      di,di                 ; see if at end
        jz      ack_hold20            ; jump if so
        mov     si,di                 ; go down the list
        jmp     ack_hold10

ack_hold20:
        mov     [si],bx               ; put behind this entry
        pop     di
        pop     si                    ; restore received message pointer
        pop     bx                    ; restore message descriptor

        ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   ACK_SEND  - Transmit next acknowledge message on the HOLDING list
;
```

```
;   NOTE: This is ONLY called during interrupt servicing when the
;   transmitter is available.  This prevents interference with the
;   IO_XMIT routine above.  (They both access the H/W)
;
;   INPUTS:
;       The PID and SEQUENCE number to acknowledge is at the head of
;       the "HOLD" queue.
;
;   Acknowledgements simply have:   DST, SRC, length, ACK_SEQ, ACK_CMD
;
ACK_Send:
        push    ax              ; save all registers
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es

        dec     word ptr [TX_READY]     ; mark transmitter as busy
;
;  Get entry off of HOLDING list
;
        mov     bx,[ACK_HOLDING]
        mov     ax,[bx]                 ; get next pointer
        mov     [ACK_HOLDING],ax        ; remove tnis ACK from holding list
        mov     si,[bx+ack_pid]         ; get processor id of originator
        mov     dx,[bx+ack_seq]         ; and sequence #
        call    ACK_DEALLOCATE          ; put on free list
;
;  Build acknowledge packet
;
        cld                             ; set auto increment
        les     di,[CARD_RAM]           ; point to hardware buffer area
        mov     cl,3                    ; 8 bytes per net table index
        shl     si,cl                   ; convert PID to net table index
        lea     si,NET_TABLE[si]        ; fetch address of dest.
        mov     cx,DEF_addr_size        ; in words
        rep     movsw                   ; copy in dest address
        mov     si,[PID]                ; get our processor id
        mov     cl,3                    ; mult by 8 (bytes/address entry)
        shl     si,cl                   ; index into address table
        lea     si,NET_TABLE[si]        ; fetch our address
        mov     cx,DEF_addr_size
        rep     movsw                   ; copy in source addr
;
;  length field is fixed to include up to command only
;
        mov     ax,DEF_pkt_cmd+2
        stosw                           ; put in buffer
```

```
;
;  Put ACKING Sequence Number in packet
;
        mov     ax,dx                   ; get acking seq number
        stosw                           ; put in buffer
        mov     ax,DEF_ACK              ; set command to ACK
        stosw                           ; stuff in buffer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  That's it for loading the packet buffer, now kick off transmission      ;
;  Setup NIC registers to begin transmission                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        mov     dx,nic_cr               ; select Page_0
        mov     al,eth_access_Page_0
        out     dx,al

        mov     dx,nic_tpsr             ; page start register
        mov     al,eth_xmit_buf_start   ; transmit page at DC00:0000
        out     dx,al
;  load length of packet
        mov     ax,min_packet           ; make it is the minimum
        mov     dx,nic_tbcr0            ; load number to transfer into H/W
        out     dx,al
        mov     dx,nic_tbcr1
        mov     al,ah
        out     dx,al
;  start transmit
        mov     dx,nic_cr
        mov     al,send         ; command to initiate transmission
        out     dx,al

        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   ACK_CHECK - Check to see if any acknowledgment requests are over
;               ACK_LIMIT clock interrupts old.  Cause shutdown if so.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       I N T E R R U P T     S E R V I C E     R O U T I N E       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   This routine is invoked by the runtime timer interrupt routine to
;   allow the network IO funtions to be checked.  As a result,
;   ALL REGISTERS except AX and DS MUST BE PRESERVED!!!
;
;   Note: because this routine is run every 5 milliseconds, it has
```

```
;   been optimized to take up little time in the typical case.  For
;   this reason, there are three exit points which are executed in
;   straight line code.


ACK_CHECK:
        mov     ax,cs
        mov     ds,ax
;
;  For each communicating processor, check watch dog timer
;
        push    si
        push    di
        push    cx
        mov     si,[WATCH_LIST]
        mov     cx,[si]                 ; get number of processor to wa-ch
        jcxz    ACK08
        add     si,2
ACK05:
        mov     di,[si]                 ; get processor ID of next to watch
        add     di,di
        dec     word ptr WATCH_DOG[di]  ; check watch_dog
        jz      ACK20                   ; FAILURE
        add     si,2
        loop    ACK05
ACK08:
        pop     cx
        pop     di
        pop     si
;
;  Now see if any outstanding messages
;
        mov     ax,[ACK_PENDING]        ; get pending list
        or      ax,ax
        jnz     ACK10                   ; if message on list, must check
        retf
ACK10:
        push    si
        mov     si,ax
        mov     ax,[ACK_TIMER]          ; get current TIME
        inc     ax
        mov     [ACK_TIMER],ax          ; update
        cmp     ax,[si+ack_count]       ; see if our TIME is UP
        jz      ACK20
        pop     si
        retf                            ; return, still time before error
ACK20:
        mov     al,TIMER_EOI            ; clear timer channel interrupt
        mov     dx,VRTIF_I8259
        out     dx,al
        mov     al,0fdH                 ; shut off everything but keyboard
```

```
        inc     dx                      ; point to mask reg
        out     dx,al
        xor     ax,ax
        push    ax
        call    VRTIF_Timestamp

        jmp     Shut_Down               ; Acknowledgment timed out!


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Data AREA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
                align   4
ISR             dw      ?               ; interrupt status register
PACKET_SIZE     dw      ?               ; packet size
CARD_RAM        dd      0dc000000h      ; address of ram buffer on enet card
RECEIVE_PTR     dw      ?               ; points to current next page to rcv
XPARM_COUNT     dw      ?               ; number of xmit params left to copy
PROFILE_PTR     dw      ?               ; current ptr to parameter profile
SEQUENCE        dw      ?               ; this processor's packet sequencer

NULL_LIST       dw      0               ; zero parameters

;
;  The following semaphore is used to provide mutual exclusion to the
;  transmit side of the Ethernet card.
;
TX_READY        dw      ?               ; semaphore count
                dw      ?               ; task value
                dw      ?               ; task value

RECEIVE_FLAG    dw      ?               ; contains buffer desc in SYNC_PHASE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  BUFFER QUEUE STRUCTURE       ;
;       record                  ;
;         BUFFER_OFFSET         ;
;         NEXT_PTR              ;
;       end record;             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

RX_BUFF_HEAD    dw      (?)
RX_BUFFER       db      num_buff dup (buff_size dup (?))
RX_BUFF_Q       dw      num_buff dup (2 dup(?))  ; (BUFFER_PTR, NEXT_DESC_PTR)

;
;  The outstanding packet acknowledgement queue contains the
;  Task id and the sequence number used when transmitting
;  each (non-acknowledgement) message.  When an acknowlegement message
;  is received, this list is checked and if the ids are found,
```

```
; they are removed.  If they are not found, the acknowledgement is
; trashed (this should not occur unless the master CPU restarts while
; a slave is still transmitting an acknowledge.  However, in future
; versions with retry implemented, multiple acknowledgements may be
; possible.
;
; Queue structure:
;
;                   *************************
;                   *  NEXT QUEUE PTR       *
;                   *************************
;                   *  TASK ID              *
;                   *************************
;                   *  SEQUENCE NUMBER      *
;                   *************************
;                   *  TIMER     COUNTER    *
;                   *************************
;
ack_delay       equ     20              ; interrupts = 5 ms each
ACK_TIMER       dw      ?

ACK_FREE        dw      ?               ; list of unused acks
ACK_PENDING     dw      ?               ; list of acks we are waiting for
ACK_HOLDING     dw      ?               ; list of acks waiting to go out

ack_size        equ     8               ; bytes per entry
; list structure
ack_next        equ     0               ; point to next in ack list
ack_pid         equ     2               ; processor ID of packet to be acked
ack_seq         equ     4               ; sequence number
ack_count       equ     6               ; counter (used to time-out ack)

ACK_RECORDS     dw      num_buff dup (ack_size dup(?))  ; NEXT, SEQ#, COUNT

cseg            ends
                end
```

```
        page    55,132
        TITLE   VRTIF - Vendor Runtime Interface Module
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; FILE: DA_VRTIF                                                        ;
; Distributed Ada - Vendor Runtime Interface                            ;
; This module provides the addresses within the                         ;
; Vendor supplied runtime for required tasking primatives.              ;
;                                                                       ;
; Copyright(C) 1989, LabTek Corporation                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                include DA_DEF.ASM

                public  VRTIF_I8259, VRTIF_vector_base
                public  VRTIF_Wait, VRTIF_Signal, VRTIF_Signal_I
                public  VRTIF_Lower_Priority

                public  VRTIF_Create_Task
                public  VRTIF_Activate_Complete
                public  VRTIF_Entry
                public  VRTIF_Rendezvous_Complete
                public  VRTIF_Accept
                public  VRTIF_Select
                public  VRTIF_PutCh
                public  VRTIF_Timestamp
                public  VRTIF_TCBTID
                public  VRTIF_APPLICATION
                public  VRTIF_Init
                public  VRTIF_task_ptr
                public  VRTIF_DS
                public  VRTIF_SELECT_REC

                extrn   Create_Task:near
                extrn   Activate_Complete:near
                extrn   Request_Entry:near
                extrn   Rendezvous_complete:near
                extrn   Select:near
                extrn   Accept:near

                extrn   PID:word

VRTIF_I8259     equ     20H         ; address of interrupt controller chip
VRTIF_TCBTID    equ     22          ; offset in TCB to priority (identifies task)
VRTIF_SELECT_REC equ    6           ; bytes per stack record for each open
                                    ; alternative in a select statement

far_jmp_instruction     equ     000EAH ; jmp intersegment
retf2_instruction       equ     0CA02H ; return intersegment pop two bytes
retf_instruction        equ     000CBH ; return far
short_call_instruction  equ     000E8H ; short call

;
```

```
; NOTE: During start up (COLD START) the vector base is that of DOS.  After
; runtime intitialization has completed it is moved to 200H.  However, this
; occurs after the network initialization, and the vectors in pla at 20H
; are moved to 200H.  Subsequent restarts are "WARM" and do not effect any
; of the interrupt vector tables.
;
VRTIF_vector_base     equ     20H        ; Initial base of vector table (DOS)

VRTIF           segment at DEF_VRTIF_ADDR

                org     0               ; this is just for convience
VRTIF_DS        label   word            ; actually in different segment

                org     970H            ; offset in RT DS
VRTIF_TASK_PTR  label   word            ; offset in DS for current task

;
;  RUNTIME TASKING CALLS ADDRESSES TO ALLOW VECTORING TO THE DISTRIBUTED
;  Ada RUNTIME
;
                org     11DAH
R1Accept        label   far             ; simple accept

                org     12C1H
R1Entry         label   far             ; simplecallentry uncond

                org     13F3H
R1RendezvousComplete  label   far       ; rendezvouscomplete

                org     1476H
R1Activated     label   far             ; activated

                org     161CH
R1CreateTask    label   far             ; createtask

                org     1D28H
R1Select        label   far             ; select

                org     2608H
VRTIF_SETPRIORITYLOWER label   far

                org     39BBH
VRTIF_PutCh     label   far             ; put_character

                org     3CCEH           ; timestamp
VRTIF_Timestamp label   far

                org     40F0H           ; patch for short calls to set priority
PATCH_40F0      label   far

                org     519BH
```

```
VRTIF_Signal_I  label   far     ; R1TESI?VI V semaphore operation (interrupt)

                org     51D0H
VRTIF_Wait      label   far     ; R1TESS?P P semaphore operation (non interrupt)

                org     51E6H
VRTIF_Signal    label   far     ; R1TESS?V V semaphore operation (non-interrupt)

VRTIF           ends



;
;   APPLICATION ENTRY ADDRESS
;
vrtif2          segment at 5374H
                org     6
VRTIF_APPLICATION label far
vrtif2          ends


cseg    segment common
        assume  cs:cseg,ds:cseg,es:cseg
        org     1100H
;
;  p. ch in calls to distributed runtime to allow runtime checking
;  o⁴ distribution.
;
VRTIr_Init:
        mov     ax,DEF_VRTIF_ADDR        ; segment for Vendor Runtime
        mov     es,ax

; Crcate Task
        lea     di,R1createtask
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Create_Task
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax

; Act,vate Complete
        lea     di,R1Activated
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Activate_Complete
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax
```

```
; entry calls
        lea     di,R1entry
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Request_Entry
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax

; end rendezvous
        lea     di,R1rendezvouscomplete
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Rendezvous_Complete
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax

; Select
        lea     di,R1Select
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Select
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax

; Accept
        lea     di,R1Accept
        mov     byte ptr es:[di],far_jmp_instruction
        inc     di
        lea     ax,Accept
        mov     es:[di],ax
        add     di,2
        mov     ax,cs
        mov     es:[di],ax
;
;   SETUP special short call transfer area at end of patch for setting
;   the priority
;
        lea     di,PATCH_40F0
        mov     byte ptr es:[di],short_call_instruction
        mov     word ptr es:[di+1],offset VRTIF_SETPRIORITYLOWER-(PATCH_40F0+3)
        mov     byte ptr es:[di+3],retf_instruction

;
:  If this is not the master, do not use PutChar... must patch
```

```
;  it out.
;
        test    CS:[PID],0FFFFh    ; master is always zero
        jz      Init_10
        lea     di,VRTIF_PutCh
        mov     word ptr es:[di],retf2_instruction
        mov     byte ptr es:[di+2],0 ; high half of count
Init_10:
        mov     ax,cs           ; restore data segment
        mov     ds,ax
        ret
;-----------------------------------------------------------------

        assume  ds:VRTIF

VRTIF_Accept:
        mov     ds,[VRTIF_DS]
        mov     si,[VRTIF_TASK_PTR]
        jmp     R1Accept+8

VRTIF_Entry:
        mov     ds,[VRTIF_DS]
        mov     si,[VRTIF_TASK_PTR]
        jmp     R1Entry+8

VRTIF_Rendezvous_Complete:
        mov     ds,[VRTIF_DS]
        mov     si,[VRTIF_TASK_PTR]
        jmp     R1rendezvouscomplete+8

VRTIF_Activate_Complete:
        mov     ds,[VRTIF_DS]
        mov     si,[VRTIF_TASK_PTR]
        jmp     R1activated+8

VRTIF_Create_Task:
        push    bp
        mov     bp,sp
        mov     ds,[VRTIF_DS]
        jmp     R1createtask+7

VRTIF_Select:
        push    bp
        mov     bp,sp
        mov     ds,[VRTIF_DS]
        jmp     R1select+7

VRTIF_Lower_Priority:
        call    Patch_40F0
        ret
```

```
cseg    ends

        end
```

```
        page    55,132
        TITLE   D,CB - Distributed Task Control Block Module
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA_DTCB.ASM                                                      ;
;                                                                          ;
;  DA - Distributed Task Control Block Module                              ;
;                                                                          ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT    USA            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                          ;
;  Ver Date     Description                                                ;
;                                                                          ;
;  0.2 Dec-89 : Enhanced to support error detection and dynamic            ;
;               configuration                                              ;
;                                                                          ;
;                                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        include DA_DEF.ASM
        .model  large

        public  DTCB_INIT
        public  SYNCHRO_SEMAPHORE
        public  CONTINUE_SEMAPHORE
        public  TASK_DIRECTORY

        public  proc_table_size
        public  REMOTE_CPU_TABLE
        public  NAME_TABLE

        public  WATCH_LIST
        public  WATCH_DOG

        extrn   MODE_SELECT:word
        extrn   PID:word

cseg    segment common

        assume  cs:cseg,ds:cseg,es:cseg
        org     0A00H
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Task Control Blocks
;       Semaphore       3 words
;       Reply Pointer   1 word
;       Return Address  2 words
;       Num Entries     1 word
;       Entry Table     N * 3 words (where N is the number of entries)
;               Profile Ptr     1 word
;               Wait Flag       1 word
;               Entry Queue     2 words
;
```

```
;       Profile List:
;               Number of Parameters    1 word
;               Mode                    1 word  ("in", "out", "in out")
;               Type/Length             1 word  (negative means unconstrained)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;    The TCB contains a synchronize semaphore which is used to
;    suspend itself and wait for a signal from another task.
;    This is followed by a reply pointer used to hold the buffer
;    descriptor of the message to which a reply is due.  Then
;    the entry information is provided.  This begins with the number
;    of entries for this task, followed by a record for each entry.
;    Each entry record contains:
;      - A profile pointer which provides the offset within the CS for
;        information on the parameter profile for this entry.
;      - A waiting flag used by the accepting task to indicate that it
;        has suspended waiting for a call on this entry (and possibly
;        others).
;      - A buffer List Pointer, This points to the buffer descriptor
;        for the first caller to this entry.  The buffer descriptor
;        provides the actual buffer address and a link to the next
;        descriptor.  This provides the FIFO queue for each entry.
;
semaphore       struc
                dw      3 dup (?)
semaphore       ends

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   TCB_INIT - Initialize Distributed Task Control Blocks
;
DTCB_Init:
        xor     ax,ax

        mov     [SYNCHRO_SEMAPHORE],ax
        mov     [SYNCHRO_SEMAPHORE+2],ax
        mov     [SYNCHRO_SEMAPHORE+4],ax

        mov     [CONTINUE_SEMAPHORE],ax
        mov     [CONTINUE_SEMAPHORE+2],ax
        mov     [CONTINUE_SEMAPHORE+4],ax

;
;   Initialize Distributed Task Directory Based on Current Operating Mode
;   This calls the Init_TCB for each task to initialize its structures.
;
INIT_DIRECTORY:
        mov     si,[MODE_SELECT]            ; fetch mode
        dec     si                          ; mode1 => offset 0
        add     si,si                       ; make word index
        mov     si,MODE_TABLE[si]           ; fetch address of mode values
```

```
        mov     cx,total_tasks

        lea     di,TASK_DIRECTORY
        mov     ax,ds
        mov     es,ax
        xor     ax,ax
        cld

INIT_DIR_LOOP:
        movsw                           ; transfer local/distrib flag
        movsw                           ; transfer PID
        mov     bx,[di]                 ; get TCB pointer
        call    Init_TCB
        add     di,2                    ; skip the distrib TCB pointer
        stosw                           ; zero counter for tasks of this type
        loop    INIT_DIR_LOOP
;
;   Initialize Watch Dog Timer Information based on configuration and
;   this processor's ID
;
        mov     ax,[PID]
        mov     bx,DEF_max_cpus*2       ; bytes per cpu entry
        mul     bx
        mov     bx,ax                   ; BX = cpu offset
        mov     ax,[MODE_SELECT]
        dec     ax                      ; adjust by one to make zero origin
        mov     dx,(DEF_max_cpus*DEF_max_cpus) * 2 ; bytes per mode table
        mul     dx
        mov     si,ax
        lea     si,WATCH_TABLE[si+bx]   ; get address of watch list entry
        mov     [WATCH_LIST],si         ; set value for other's use
        mov     cx,[si]                 ; fetch number of timers to init
        jcxz    Watch_init_done
        add     si,2
Watch_Init:
        mov     bx,[si]                 ; fetch PID of processor to watch
        add     bx,bx
        mov     WATCH_DOG[bx],DEF_WATCH_DOG_LIMIT ; init timer for this pid
        add     si,2
        loop    Watch_init
Watch_init_done:
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Init_TCB : zero all semaphore words and entry table values for the TCB
;   pointed to by:
;                       INPUT:  BX points to TCB of interest
;                               AX contains zero
Init_TCB:
        or      bx,bx           ; does this task have a TCB?
```

```
        jz      Init_TCB_30      ; exit if not applicable
        mov     [bx],ax          ; clear semaphore
        mov     [bx+2],ax
        mov     [bx+4],ax
        push    cx
        mov     cx,[bx+DEF_num_entries]      ; fetch number of entries
        lea     bx,DEF_entry_table[bx]
        jcxz    Init_TCB_20                  ; if no entries
Init_TCB_10:
        mov     DEF_entry_wait[bx],ax        ; zero wait flag
        mov     DEF_entry_queue[bx],ax       ; zero buffer descriptor
        mov     DEF_entry_queue+2[bx],ax     ; zero next pointer
        add     bx,size DEF_entry_rec        ; go to next record
        loop    Init_TCB_10
Init_TCB_20:
        pop     cx
Init_TCB_30:
        ret

        align   4

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Configuration Mode Control
;
;   THESE TABLES WOULD NORMALLY BE PRODUCED BY A CONFIGURATION CONTROL
;   TOOL, BUT FOR PROTOTYPE PURPOSES THEY ARE GENERATED BY HAND.
;
;   The current configuration control allows for four different
;   operating modes and three processors (alpha, bravo, and charlie):
;
;     MODE 1 : All tasks are on the alpha processor
;     MODE 2 : All BDS tasks are on alpha, all simulator tasks are on bravo.
;     MODE 3 : All BDS tasks are on alpha, all simulator tasks are on charlie.
;     MODE 4 : All BDS tasks except one of the guidance tasks is on alpha,
;              the simulator is on bravo, and one guidance task is on charlie
;
;   The mode (below) is initialized during system startup.  The master
;   processor asks the operator which mode to use.  If a system failure
;   occurs, the master shuts down the system and brings it back up as
;   a single processor version.  Note that a system function:
;
;   Configuration Table - for each task, the location is defined in terms
;   of current operating mode.
;
MODE_TABLE      label   word
        dw      offset  MODE1
        dw      offset  MODE2
        dw      offset  MODE3
        dw      offset  MODE4
```

```
;
;  TASK LOCATION DIRECTORY because of lack of compiler
;  support, very little information is available to uniquely correlate
;  tasks during runtime calls.  As a workaround, unique priorities
;  are used for each task type, and counters are supplied for multiple
;  tasks within the type which modify the identification with respect
;  to the task priority.  In this way, each task can be quickly correlated
;  to its distributed characteristics at runtime.
;
;  The following directory contains entries for each task and is indexed
;  by task priority.  The entries are:
;
;           <LOCAL/DISTRIBUTED FLAG> <PID> <DIST_TCB_PTR> and <spare>
;
;  The LOCAL/DISTRIBUTED FLAG indicates if all entry calls are local.  If
;  one is distributed, they must all go through the distributed runtime,
;  even if the call being made is local.  <PID> is the processor that the
;  task is resident on.  For calls being made through the distributed
;  runtime, additional task control information is located by the pointer
;  DIST_TCB_PTR.
;
;  This directory is initialized during configuration time based on
;  operator or automatic mode selection.  The first two values are
;  set according to mode, the last two are statically defined.
;
;*******************************************************************************
TASK_DIRECTORY label   word
        dw      4 dup (0)                               ; dummy to offset  32
        dw      ?,?,        SAVE_TCB,   ?               ;(12)save          31
        dw      ?,?,     DISPLAY_TCB,   ?               ;(11)display       30
        dw      ?,?,   TRACK_DAT_TCB,   ?               ;(10)track_data    29
        dw      ?,?,   REPORTBUF_TCB,   ?               ;(09)report_buf    28
        dw      ?,?,    GUIDEBUF_TCB,   ?               ;(08)guide_buf     27
        dw      ?,?,     ROCKSUP_TCB,   ?               ;(07)rock_sup      26
        dw      ?,?,     TARGSUP_TCB,   ?               ;(06)targ_sup      25
        dw      ?,?,     CONTROL_TCB,   ?               ;(05)control       24
        dw      ?,?,   GUIDANCE1_TCB,   ?               ;(04)guidance(1)   23
        dw      ?,?,   GUIDANCE2_TCB,   ?               ;(03)guidance(2)   22
        dw      ?,?,       TRACK_TCB,   ?               ;(02)track         21
        dw      ?,?,      UPDATE_TCB,   ?               ;(01)update        20
        dw      ?,?,        MAIN_TCB,   ?               ;(00)bds           19
total_tasks     equ     ($-TASK_DIRECTORY)/8 ; must follow definitions above
;*******************************************************************************

local_entries   equu    0
dist_entries    equ     1


;
;   For each mode (of four) the local/distrib flag must be set and the pid
;   must be set.
```

```
;
MODE1   label   word
;               DISTRIBUTED  ,  PID
;               ------------    -----
        dw      2 dup (0)                   ; dummy to offset priority by one
        dw      local_entries,  DEF_alpha          ;(12)save
        dw      local_entries,  DEF_alpha          ;(11)display
        dw      local_entries,  DEF_alpha          ;(10)track_data
        dw      local_entries,  DEF_alpha          ;(09)report_buf
        dw      local_entries,  DEF_alpha          ;(08)guide_buf
        dw      local_entries,  DEF_alpha          ;(07)rock_sup
        dw      local_entries,  DEF_alpha          ;(06)targ_sup
        dw      local_entries,  DEF_alpha          ;(05)control
        dw      local_entries,  DEF_alpha          ;(04)guidance(1)
        dw                  0,     DEF_NA          ;(03)guidance(2)
        dw      local_entries,  DEF_alpha          ;(02)track
        dw      local_entries,  DEF_alpha          ;(01)update
        dw      local_entries,  DEF_alpha          ;(00)bds

MODE2   label   word
;               DISTRIBUTED  ,  PID
;               ------------    -----
        dw      2 dup (0)                   ; dummy to offset priority by one
        dw      local_entries,  DEF_alpha          ;(12)save
        dw      local_entries,  DEF_alpha          ;(11)display
        dw      local_entries,  DEF_alpha          ;(10)track_data
        dw      dist_entries,   DEF_bravo          ;(09)report_buf
        dw      dist_entries,   DEF_bravo          ;(08)guide_buf
        dw      local_entries,  DEF_bravo          ;(07)rock_sup
        dw      dist_entries,   DEF_bravo          ;(06)targ_sup
        dw      local_entries,  DEF_alpha          ;(05)control
        dw      local_entries,  DEF_alpha          ;(04)guidance(1)
        dw                  0,     DEF_NA          ;(03)guidance(2)
        dw      local_entries,  DEF_alpha          ;(02)track
        dw      local_entries,  DEF_alpha          ;(01)update
        dw      local_entries,  DEF_alpha          ;(00)bds

MODE3   label   word
;               DISTRIBUTED  ,  PID
;               ------------    -----
        dw      2 dup (0)                   ; dummy to offset priority by one
        dw      local_entries,  DEF_alpha          ;(12)save
        dw      local_entries,  DEF_alpha          ;(11)display
        dw      local_entries,  DEF_alpha          ;(10)track_data
        dw      dist_entries,   DEF_charlie        ;(09)report_buf
        dw      dist_entries,   DEF_charlie        ;(08)guide_buf
        dw      local_entries,  DEF_charlie        ;(07)rock_sup
        dw      dist_entries,   DEF_charlie        ;(06)targ_sup
        dw      local_entries,  DEF_alpha          ;(05)control
        dw      local_entries,  DEF_alpha          ;(04)guidance(1)
```

```
        dw                 0,      DEF_NA          ;(03)guidance(2)
        dw      local_entries,  DEF_alpha          ;(02)track
        dw      local_entries,  DEF_alpha          ;(01)update
        dw      local_entries,  DEF_alpha          ;(00)bds

MODE4   label   word
;                DISTRIBUTED  ,  PID
;               -------------   -----
        dw      2 dup (0)                  ; dummy to offset priority by one
        dw      local_entries,  DEF_alpha          ;(12)save
        dw      local_entries,  DEF_alpha          ;(11)display
        dw      local_entries,  DEF_alpha          ;(10)track_data
        dw      dist_entries,   DEF_bravo          ;(09)report_buf
        dw      dist_entries,   DEF_bravo          ;(08)guide_buf
        dw      local_entries,  DEF_bravo          ;(07)rock_sup
        dw      dist_entries,   DEF_bravo          ;(06)targ_sup
        dw      local_entries,  DEF_alpha          ;(05)control
        dw      dist_entries,   DEF_charlie        ;(04)guidance(1)
        dw      local_entries,  DEF_alpha          ;(03)guidance(2)
        dw      local_entries,  DEF_alpha          ;(02)track
        dw      local_entries,  DEF_alpha          ;(01)update
        dw      local_entries,  DEF_alpha          ;(00)bds


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Task Control Blocks
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;------------------------------------------------------------------------

Main_TCB        semaphore       <>
                dw      ?       ; Reply Pointer
                dw      2 dup (?) ; Return Address
                dw      0       ; Number of Entries

;------------------------------------------------------------------------
Targsup_TCB     SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      1       ; num of distributed entries
                DEF_entry_rec   <offset Next_Target_Msg,?,?,?>
    Next_Target_Msg     dw      1       ; 1 parameter = TARGET_MSG_TYPE
                        dw      DEF_OUT ; mode = out
                        dw      802     ; only allow 50 targets for now!

;------------------------------------------------------------------------
Rocksup_TCB     SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;------------------------------------------------------------------------
```

```
Guidebuf_TCB    SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      2       ; num of distributed entries
                DEF_entry_rec   <offset Put_Guide,?,?,?>
                DEF_entry_rec   <offset Get_Guide,?,?,?>

    Put_Guide           dw      1        ; 1 parameter
                        dw      DEF_IN   ; in mode
                        dw      122      ; # bytes

    Get_Guide           dw      1        ; 1 parameter
                        dw      DEF_OUT  ; out mode
                        dw      122      ; # bytes

;----------------------------------------------------------------------------
Reportbuf_TCB   SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      2       ; num of distributed entries
                DEF_entry_rec   <offset Put_Report,?,?,?>
                DEF_entry_rec   <offset Get_Report,?,?,?>

    Put_Report          dw      1        ; 1 parameter
                        dw      DEF_IN   ; in mode
                        dw      322      ; # bytes

    Get_Report          dw      1        ; 1 parameter
                        dw      DEF_OUT  ; out mode
                        dw      322      ; # bytes

;----------------------------------------------------------------------------
Track_TCB       SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;----------------------------------------------------------------------------
Control_TCB     SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;----------------------------------------------------------------------------
Guidance1_TCB   SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      2       ; num of distributed entries
                DEF_entry_rec   <offset History,?,?,?>
                DEF_entry_rec   <offset Next_Guidance,?,?,?>
    History             dw      1        ; 1 parameter
```

```
                        dw      DEF_IN  ; mode is in
                        dw      -1      ; unconstrained
   Next_Guidance        dw      1       ; 1 parameter
                        dw      DEF_OUT ; mode is out
                        dw      -1      ; unconstrained

;----------------------------------------------------------------------------
Guidance2_TCB   SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      2       ; num of distributed entries
                DEF_entry_rec   <offset History,?,?,?>
                DEF_entry_rec   <offset Next_Guidance,?,?,?>

;----------------------------------------------------------------------------
SAVE_TCB        SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;----------------------------------------------------------------------------
DISPLAY_TCB     SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;----------------------------------------------------------------------------
TRACK_DAT_TCB   SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;----------------------------------------------------------------------------
UPDATE_TCB      SEMAPHORE       <>
                dw      ?       ; reply
                dw      2 dup (?) ; Return Address
                dw      0       ; num of distributed entries

;
; These semaphores are used for synchronization of tasks among all
; processors during program startup.
;
SYNCHRO_SEMAPHORE       dw      3 dup (?)
CONTINUE_SEMAPHORE      dw      3 dup (?)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  PROCESSOR TABLE: Given the Mode, the number and pids are provided ;
;         fields:       # of remote CPU's, CPU1-ID, CPU2-ID          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
proc_table_size equ     6               ; bytes per entry
REMOTE_CPU_TABLE label  word
```

```
                dw      0, DEF_NA,      DEF_NA
                dw      1, DEF_bravo,   DEF_NA
                dw      1, DEF_charlie, DEF_NA
                dw      2, DEF_bravo,   DEF_charlie

NAME_TABLE      label   word
        dw      offset  ALPHA_NAME
        dw      offset  BRAVO_NAME
        dw      offset  CHARLIE_NAME

ALPHA_NAME      db      'Alpha',0
BRAVO_NAME      db      'Bravo',0
CHARLIE_NAME    db      'Charlie',0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Watch Dog Timer Data : These Structures determine which processors ;
;  to monitor for activity as a function of processor ID and mode.    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        align   2

;  The following table contains a block for each mode, which contains
;  an entry for each processor.  Each entry contains a count, followed
;  by the PIDs to watch.  This table defines which processors communicate
;  with each other during the various modes.
;
Watch_table_entry_size  equ     8
WATCH_TABLE     label   word
;MODE1              -- no processors to watch
          dw      0,DEF_NA,   DEF_NA    ; PID 0
          dw      0,DEF_NA,   DEF_NA    ; PID 1
          dw      0,DEF_NA,   DEF_NA    ; PID 2
;MODE2
          dw      1,DEF_bravo,DEF_NA    ; PID 0
          dw      1,DEF_alpha,DEF_NA    ; PID 1
          dw      0,DEF_NA,   DEF_NA    ; PID 2
;MODE3
          dw      1,DEF_charlie, DEF_NA   ; PID 0
          dw      0,DEF_NA,      DEF_NA   ; PID 1
          dw      1,DEF_alpha,   DEF_NA   ; PID 2
;MODE4
          dw      2,DEF_bravo,   DEF_charlie      ; PID 0
          dw      1,DEF_alpha,   DEF_NA           ; PID 1
          dw      1,DEF_alpha,   DEF_NA           ; PID 2

WATCH_LIST      dw      ?               ; points to list for current config
WATCH_DOG       dw      DEF_max_cpus dup (?)  ; table of timers


cseg      ends

        end
```

THIS PAGE INTENTIONALLY LEFT BLANK.

```
                page    55,132
                TITLE   Setup - Distributed Ada Network Initialization
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA_SETUP.ASM                                                   ;
;  Distributed Ada - Setup                                               ;
;                                                                        ;
;  This module initilizes the network to prepare for distributed         ;
;  processing.                                                           ;
;                                                                        ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT. USA            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                .model  large
                public  Setup
                public  PID           ; processor ID
                public  NET_TABLE     ; addresses indexed 8 per PID

                include DA_HW.ASM
cseg            segment     common
                assume  cs:cseg,ds:cseg,es:cseg
                org     1C00H

Setup:
                mov     dx,cntrl           ; Gate array controller
                mov     al,eth_enable_reset
                out     dx,al
                mov     al,eth_disable_reset
                out     dx,al
                mov     al,eth_access_prom
                out     dx,al
                mov     cx,6
                mov     ax,cs
                mov     es,ax                       ; set es:di to receive board
                mov     di,offset BOARD_ADDRESS     ; address from prom
                mov     dx,prom_address_0
                cld
GET_ADDRESS:
                in      al,dx
                stosb
                inc     dx
                loop    GET_ADDRESS

                mov     dx,cntrl                    ; select no-sharing adapter,
                mov     al,eth_recv_select          ; and external transceiver
                out     dx,al

                mov     dx,gacfr                    ; 8K of memory mapped space,
                mov     al,eth_lan_config           ; with interrupts enabled
                out     dx,al

                mov     dx,dqtr                     ; # of bytes to transfer on
                mov     al,eth_rem_DMA_burst        ; a remote DMA burst (n/a)
```

```
out   dx,al

mov   dx,idcfr                ; interrupt IRQ and DMA
mov   al,eth_irq_line         ; channel selection (DMA n/a)
out   dx,al

mov   dx,damsb                ; 8k configuration for remote
mov   al,eth_rem_DMA_config   ; DMA. Not used, but minimum
out   dx,al                   ; value needed

mov   dx,pstr                 ; start of receive buffer.
mov   al,eth_recv_buf_start   ; Value MUST match that in
out   dx,al                   ; NIC_pstart

mov   dx,pspr                 ; end of receive buffer.
mov   al,eth_recv_buf_end     ; Value MUST match that in
out   dx,al                   ; NIC_pstop

mov   dx,NIC_cr               ; stop NIC activity
mov   al,eth_nic_stop
out   dx,al

mov   dx,NIC_dcr              ; local DMA transfers as
mov   al,eth_nic_DMA_config   ; 8 byte bursts
out   dx,al

mov   dx,NIC_rbcr0            ; remote DMA setup (remote
mov   al,eth_remote_DMA_lo    ; DMA not used, only local
out   dx,al                   ; used)

mov   dx,NIC_rbcr1            ; hi byte of # of bytes to
mov   al,eth_remote_DMA_hi    ; transfer during a remote
out   dx,al                   ; DMA operation

mov   dx,NIC_rcr              ; accept only good packets
mov   al,eth_packet_types     ;
out   dx,al

mov   dx,NIC_tcr              ; go into internal loopback
mov   al,eth_nic_mode         ; mode to finish programming
out   dx,al                   ; (see anomalies - p. 52)

mov   dx,NIC_bndy             ; overwrite protection rgtr.
mov   al,eth_bndy_start       ; (protects unread packets)
out   dx,al

mov   dx,NIC_pstart           ; start of receive queue
mov   al,eth_recv_buf_start
out   dx,al

mov   dx,NIC_pstop            ; end of receive queue
```

```
                mov   al,eth_recv_buf_end
                out   dx,al

                mov   dx,NIC_isr              ; clear interrupt status
                mov   al,eth_int_status
                out   dx,al

                mov   dx,NIC_imr              ; disable interrupts
                mov   al,eth_ints_disabled    ; for receive and xmit
                out   dx,al                   ;

                mov   dx,NIC_cr               ; access page 1 registers
                mov   al,eth_access_page_1
                out   dx,al

                mov   dx,phys_address_0        ; let NIC know its address
                mov   ax,cs
                mov   ds,ax
                mov   si,offset BOARD_ADDRESS  ; from the prom
                cld
                mov   cx,6                     ; number of addresses to give
GIVE_ADDRESS:
                lodsb
                out   dx,al
                inc   dx
                loop  GIVE_ADDRESS            ; load all addresses

                mov   dx,NIC_curr             ; load current receive pointer
                mov   al,eth_recv_buf_start   ; with pstart
                out   dx,al

                mov   dx,NIC_cr               ; access page 0 registers
                mov   al,eth_access_page_0
                out   dx,al

                mov   dx,NIC_cr               ; start NIC chip
                mov   al,eth_start_nic
                out   dx,al

                mov   dx,NIC_tcr              ; exit internal loopback mode
                mov   al,eth_exit_mode
                out   dx,al
;
; Note: The RAM initialization is necessary for the multi-packet processing
; done in the receive interrupt routine
;
                mov   ax,net_memory_seg       ; initialize LAN memory to
                mov   es,ax                   ; zeroes
                mov   cx,net_memory_size/2    ; in words
                xor   di,di                   ; start at begin of segment
                cld
```

```
                mov   ax,0000                  ; initialization value
FILL:
                stosw
                loop  FILL

;
;  Now check our address against the known Ethernet addresses to determine
;  our processor ID
;
                mov  ax,cs
                mov  es,ax                  ; ds already = cs
                mov  bx,0                   ; init processor ID
                mov  di,offset NET_TABLE
                cld                         ; search direction = increment
Search:
                push di                     ; save start of current net addr
                mov  cx,3                   ; three words per address
                mov  si,offset BOARD_ADDRESS
                repe cmpsw
                pop  di                     ; restore current table index
                jz   Found
                add  di,8                   ; go to next index
                inc  bx                     ; count processor  id
                cmp  bx,NET_COUNT           ; see if all searched
                jnz  Search                 ; loop back if more
;
;  If not found, it will return processor id = NET_COUNT
;
Found:
                mov  [PID],BX               ; record Processor ID
                ret                         ; done with Setup

                align 2
;
;  VALID PROCESSOR ID's Determined by Ethernet ADDRESS
;           0 - ALPHA
;           1 - BRAVO
;           2 - CHARLIE
;
PID             dw    ?                      ; Processor ID

BOARD_ADDRESS   db    6 dup (?)              ; holds board address

;
;   PROCESSOR STATION ADDRESS TABLE
;
NET_COUNT       equ    6                    ; number of processor on net

NET_TABLE       label  byte
      db     02H, 60H, 8CH, 47H, 61H, 82H,0,0  ; processor Alpha    0 EARTH
      db     02H, 60H, 8CH, 47H, 63H, 55H,0,0  ; processor Bravo    1 VENUS
```

```
        db      02H, 60H, 8CH, 48H, 51H, 60H,0,0  ; processor Charlie  2
        db      02H, 60H, 8CH, 58H, 35H, 68H,0,0  ; processor Delta    3
        db      02H, 60H, 8CH, 02H, 00H, 58H,0,0  ; processor Echo     4
        db      02H, 60H, 8CH, 44H, 52H, 09H,0,0  ; processor Foxtrot  5

cseg                    ends

                        END
```

```
        page    55,132
        TITLE   Sync - Distributed Ada Network Synchronization
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA_SYNC.ASM                                                     ;
;  Distributed Ada - Setup                                                ;
;                                                                         ;
;  This procedure varies depending on the processor type (master/slave) ;
;  and the operational phase (Sync_Phase vs. normal).  During           ;
;  SYNC_PHASE, the vendor runtime is not used at all (ie. no tasking)   ;
;  and a wait loop is used to detect incoming packets.  Since it is     ;
;  likely that messages will be lost during SYNC_PHASE, a different     ;
;  protocol is used which does not specifically utilize ACK messages.   ;
;  Instead, resend and long time-outs are used to synchronize.  A       ;
;  "Cold_Start" command is used here to definitively restart the system.;
;                                                                         ;
;  During normal operation, the master sends a "sync_start" and waits   ;
;  for a "sync_ready" from each slave.  Then it sends a "sync_continue" ;
;  to continue with processing (all processors have syncrhonized).      ;
;                                                                         ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT. USA            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        .model  large

        include DA_DEF.ASM

        public  Sync
;                                       MODULE
        extrn   MASTER:word           ;   (da)
        extrn   SYNC_PHASE:word       ;   (da)
        extrn   NUM_ROCKETS:word      ;   (da)
        extrn   NUM_TARGETS:word      ;   (da)
        extrn   MODE_SELECT:word      ;   (da)
        extrn   Print:near            ;   (da)
        extrn   Shut_down:near        ;   (da)

        extrn   SYNCHRO_SEMAPHORE:word  ;(dtcb)
        extrn   CONTINUF_SEMAPHORE:word ;(dtcb)

        extrn   RECEIVE_FLAG:word       ;(io)
        extrn   TX_READY:word           ;(io)
        extrn   IO_Deallocate:near      ;(io)
        extrn   IO_Xmit:near            ;(io)

        extrn   VRTIF_WAIT:far          ;(vrtif)

        extrn   proc_table_size:abs
        extrn   REMOTE_CPU_TABLE:word
        extrn   NAME_TABLE:word

cseg    segment     common
        assume  cs:cseg,ds:cseg,es:cseg
```

```
        org     2100H

Sync:
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es
        test    [MASTER],0FFFFH         ; are we a master?
        jnz     Sync10                  ; jmp if master
        jmp     slave                   ; no, go act like a slave
Sync10:
        test    [SYNC_PHASE],0FFFFH     ; are we in sync phase
        jnz     Master_Sync
        jmp     Master_Normal

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; MASTER SYNC PHASE:
;
; synchronize with slave processors and send them configuration
; information.
;
Master_Sync:
        mov     [MASTER_SYNC_DATA_PTR],cs ; first setu  Parameter Data Pointer
        mov     ax,[NUM_ROCKETS]        ; load Configu .io.. Record
        mov     [CONFIG.ROCKETS],ax
        mov     ax,[NUM_TARGETS]
        mov     [CONFIG.TARGETS],ax
        mov     ax,[MODE_SELECT]
        mov     [CONFIG.SELECT],ax
        dec     ax                      ; adjust mode1 => 0

        mov     [RETRY_COUNT],DEF_retry_times   ; initialize retry counter
        mov     dx,proc_table_size      ; number of bytes per entry (ax=mode)
        mul     dx                      ; compute address
        lea     si,REMOTE_CPU_TABLE     ;
        add     si,ax                   ; index to proper selected mode
        mov     [REMOTE_INDEX],si       ; save index into table
        mov     cx,[si]                 ; fetch number of processors
        mov     [CPU_COUNT],cx          ; remaining CPUS to process
        add     si,2                    ; skip of number
        mov     [CPU_PTR],si            ; save pointer to current CPU
        or      cx,cx
        jnz     MSP10                   ; if there are some remote CPUs
        jmp     Sync90                  ; if none

MSP10:
;
```

```
        lea     si,crlf
        call    Print

        jmp     Shut_down
;
;  Sync occured, print notification and Go on to next processor in list
;
MSP30:
        mov     bx,[RECEIVE_FLAG]       ; get buffer pointer
        call    IO_Deallocate           ; return buffer

        lea     si,Success
        call    Print
        lea     si,crlf
        call    Print

        mov     si,[CPU_PTR]            ; get CPU pointer
        add     si,2
        mov     [CPU_PTR],si            ; update
        dec     [CPU_COUNT]             ; count down
        jz      MSP35                   ; continue if done with loop
        jmp     MSP10                   ; otherwise loop back
;
;  Now All processors have "checked in".  Send them each a "continue"
;
MSP35:
        lea     si,Sync_Complete
        call    Print

        mov     si,[REMOTE_INDEX]       ; get index into table back
        mov     cx,[si]                 ; fetch number of processors
        mov     [CPU_COUNT],cx          ; remaining CPUS to process
        add     si,2                    ; skip of number
        mov     [CPU_PTR],si            ; save as current CPU pointer
MSP40:
        test    [TX_READY],0FFFFH       ; make sure the transmitter is free
        jle     MSP40                   ; wait if not

        sub     sp,6                    ; skip parameter stuff
        lea     ax,MASTER_CONTINUE_PROFILE  ; profile
        push    ax
        sub     sp,6                    ; skip MY_TID, ENTRY, and TID
        mov     ax,DEF_sync_continue    ; command
        push    ax
        push    [si]                    ; processor ID of destination
        call    IO_Xmit                 ; send message
        mov     si,[CPU_PTR]            ; get CPU pointer
        add     si,2
        mov     [CPU_PTR],si            ; update
        dec     [CPU_COUNT]             ; count down
        jnz     MSP40
```

```
;
; Wait for last transmit complete interrupt
;
MSP50:
        test    [TX_READY],0FFFFH       ; make sure the transmitter is free
        jle     MSP50                   ; wait if not
        jmp     Sync90                  ; done

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; MASTER NORMAL PHASE : runtime synchronization after configuration setup
;
; just synchronize with slave processors
;
Master_Normal:
        mov     ax,[MODE_SELECT]
        dec     ax                      ; mode1 => 0
        mov     dx,proc_table_size      ; number of bytes per entry (ax=mode)
        mul     dx                      ; compute address
        lea     si,REMOTE_CPU_TABLE     ;
        add     si,ax                   ; index to proper selected mode
        mov     [REMOTE_INDEX],si       ; save index into table
        mov     cx,[si]                 ; fetch number of processors
        mov     [CPU_COUNT],cx          ; remaining CPUS to process
        add     si,2                    ; skip of number
        mov     [CPU_PTR],si            ; save pointer to current CPU
        or      cx,cx
        jnz     MNP10                   ; if there are some remote CPUs
        jmp     Sync90                  ; if none
MNP10:
        sub     sp,6                    ; skip parameter stuff
        lea     ax,MASTER_CONTINUE_PROFILE ; no parameters
        push    ax
                                        ; My TID
        sub     sp,6                    ; skip ENTRY, and TID
        mov     ax,DEF_sync_start       ; command
        push    ax
        mov     si,[CPU_PTR]            ; get cpu pointer back
        push    [si]                    ; processor ID of destination
        call    IO_Xmit                 ; send message
;
; Now wait for a reply
;
        push    cs
        lea     ax,SYNCHRO_SEMAPHORE
        push    ax
        call    VRTIF_Wait      ; do a wait
;
; Sync occured, Go on to next processor in list
;
MNP30:
        mov     si,[CPU_PTR]            ; get CPU pointer
```

```
        add     si,2
        mov     [CPU_PTR],si            ; update
        dec     [CPU_COUNT]             ; count down
        jz      MNP35                   ; continue if done with loop
        jmp     MNP10                   ; otherwise loop back
;
;  Now All processors have "checked in".  Send them each a "continue"
;
MNP35:
        mov     si,[REMOTE_INDEX]       ; get index into table back
        mov     cx,[si]                 ; fetch number of processors
        mov     [CPU_COUNT],cx          ; remaining CPUS to process
        add     si,2                    ; skip of number
        mov     [CPU_PTR],si            ; save as current CPU pointer
MNP40:
        sub     sp,6                    ; skip parameter stuff
        lea     ax,MASTER_CONTINUE_PROFILE  ; profile
        push    ax
        sub     sp,6                    ; skip MY_TID, ENTRY, and TID
        mov     ax,DEF_sync_continue    ; command
        push    ax
        push    [si]                    ; processor ID of destination
        call    IO_Xmit                 ; send message

        mov     si,[CPU_PTR]            ; get CPU pointer
        add     si,2
        mov     [CPU_PTR],si            ; update
        dec     [CPU_COUNT]             ; count down
        jnz     MNP40

        jmp     Sync90                  ; DONE


Slave:
        test    [SYNC_PHASE],0FFFFH     ; see if initial sync phase
        jnz     Slave05
        jmp     Slave_Normal
Slave05:
        lea     si,Slave_sync
        call    Print
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       SLAVE SYNC MODE
;
;       Wait for configuration information
Slave10:
        test    [RECEIVE_FLAG],0FFFFH   ; see if incoming data
        jz      Slave10
;
;  We got a message, check the command for "start"
;
        mov     bx,[RECEIVE_FLAG]
```

```
        mov     [RECEIVE_FLAG],0        ; zero for next time
        mov     si,[bx]                 ; fetch buffer ptr
        cmp     word ptr [si+DEF_pkt_cmd],DEF_cold_start ; is this a START
        jz      Slave20
;
;   Must be some other traffic, ignore it!
;
        call    IO_Deallocate           ; free up buffer
        jmp     Slave10

;
;   Got a valid Cold Start... respond!
;
Slave20:
        lea     si,[si+Def_pkt_data]    ; point to data area of packet buffer
        mov     ax,[si+ROCKETS]         ; unload Configuration Information
        mov     [NUM_ROCKETS],ax
        mov     ax,[si+TARGETS]
        mov     [NUM_TARGETS],ax
        mov     ax,[si+SELECT]
        mov     [MODE_SELECT],ax
        call    IO_Deallocate

Slave30:
        test    [TX_READY],0FFFFH       ; make sure the transmitter is free
        jle     Slave30                 ; wait if not

        sub     sp,6                    ; skip parameter stuff
        lea     ax,SLAVE_READY_PROFILE  ; profile
        push    ax
        sub     sp,6                    ; skip MY_TID, ENTRY, and TID
        mov     ax,DEF_sync_ready       ; command
        push    ax
        xor     ax,ax                   ; PID of master is always zero
        push    ax
        call    IO_Xmit                 ; send message
        lea     si,Success
        call    Print
;
;   Now wait for Continue
;
Slave40:
        test    [RECEIVE_FLAG],0FFFFH   ; see if incoming data
        jz      Slave40
;
;   We got a message, make sure it is continue
;
        mov     bx,[RECEIVE_FLAG]
        mov     [RECEIVE_FLAG],0        ; clear for next time
        mov     si,[bx]                 ; get buffer pointer
        cmp     word ptr [si+DEF_pkt_cmd],DEF_sync_continue ; is this a CONTINUE?
```

```
        jz      Slave50
;
;   Must be some other traffic, deallocate buffer, and check for another cold
;   start.
;
        call    IO_Deallocate           ; free up buffer
        cmp     word ptr [si+DEF_pkt_cmd],DEF_cold_start   ; COLD START?
        jnz     Slave40                 ; if not, simply ignore it
        jmp     Slave10                 ; if so, start all over
;
;  Slave synchronization has completed, deallocate buffer and exit
;
Slave50:
        call    IO_Deallocate           ; free up buffer
        jmp     Sync90

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       SLAVE NORMAL MODE
;
;       Wait for "Start" semaphore, issue "ready" then wait for "continue"
;

Slave_Normal:
        push    cs              ; push address of Semaphore
        lea     ax,SYNCHRO_SEMAPHORE
        push    ax
        call    VRTIF_Wait      ; do a wait
;
;   issue a "ready" message
;
        sub     sp,6                    ; skip parameter stuff
        lea     ax,SLAVE_READY_PROFILE  ; profile
        push    ax
        sub     sp,6                    ; skip MY_TID, ENTRY, and TID
        mov     ax,DEF_sync_ready       ; command
        push    ax
        xor     ax,ax                   ; PID of master is always zero
        push    ax
        call    IO_Xmit                 ; send message
        push    cs
        lea     ax,CONTINUE_SEMAPHORE
        push    ax              ; wait for the "go ahead"
        call    VRTIF_Wait
                                ; all done
Sync90:
        mov     [SYNC_PHASE],0  ; NO longer in Sync Phase!
        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
```

```
        pop     cx
        pop     bx
        pop     ax
        ret

        align   4

MASTER_SYNC_PROFILE     dw      1               ; provide 1 param: config record
                        dw      DEF_IN          ; mode in
                        dw      6               ; number of bytes in record

MASTER_SYNC_DATA_PTR    dw      ?               ; segment address
                        dw      offset CONFIG

MASTER_CONTINUE_PROFILE dw      0               ; no parameters
SLAVE_READY_PROFILE     dw      0               ; no parameters

CONFIG_RECORD   struc
ROCKETS         dw      ?               ; NUM_ROCKETS
TARGETS         dw      ?               ; NUM_TARGETS
SELECT          dw      ?               ; MODE_SELECT
CONFIG_RECORD   ends

CONFIG          CONFIG_RECORD   <>

RETRY_COUNT     dw      ?

CPU_COUNT       dw      ?
CPU_PTR         dw      ?
REMOTE_INDEX    dw      ?

Attempt         db      13,10,'Trying To Sync With: ',0
Failure         db      ' : Synchronization Failed',0
Success         db      ' : Synchronization Succeeded',0
Sync_Complete   db      13,10,'SYNCHRONIZATION COMPLETED',13,10,0
crlf            db      13,10,0
Period          db      ' . ',0
Slave_sync      db      'Slave Mode, Trying to Synchonize...',0

cseg    ends

        END
```

```
                page    55,132
                TITLE   Setup - Distributed Ada Network Initialization
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; FILE:  DA_SETUP.ASM                                                     ;
; Distributed Ada - Setup                                                 ;
;                                                                         ;
; This module initilizes the network to prepare for distributed          ;
; processing.                                                             ;
;                                                                         ;
; Copyright(C) 1989, LabTek Corporation, Woodbridge, CT. USA             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                .model  large
                public  Setup
                public  PID          ; processor ID
                public  NET_TABLE    ; addresses indexed 8 per PID

                include DA_HW.ASM
cseg            segment     common
                assume  cs:cseg,ds:cseg,es:cseg
                org   1C00H

Setup:
                mov   dx,cntrl          ; Gate array controller
                mov   al,eth_enable_reset
                out   dx,al
                mov   al,eth_disable_reset
                out   dx,al
                mov   al,eth_access_prom
                out   dx,al
                mov   cx,6
                mov   ax,cs
                mov   es,ax                  ; set es:di to receive board
                mov   di,offset BOARD_ADDRESS ; address from prom
                mov   dx,prom_address_0
                cld
GET_ADDRESS:
                in    al,dx
                stosb
                inc   dx
                loop  GET_ADDRESS

                mov   dx,cntrl               ; select no-sharing adapter,
                mov   al,eth_recv_select     ; and external transceiver
                out   dx,al

                mov   dx,gacfr               ; 8K of memory mapped space,
                mov   al,eth_lan_config      ; with interrupts enabled
                out   dx,al

                mov   dx,dqtr                ; # of bytes to transfer on
                mov   al,eth_rem_DMA_burst   ; a remote DMA burst (n/a)
```

```
out   dx,al

mov   dx,idcfr                ; interrupt IRQ and DMA
mov   al,eth_irq_line         ; channel selection (DMA n/a)
out   dx,al

mov   dx,damsb                ; 8k configuration for remote
mov   al,eth_rem_DMA_config   ; DMA. Not used, but minimum
out   dx,al                   ; value needed

mov   dx,pstr                 ; start of receive buffer.
mov   al,eth_recv_buf_start   ; Value MUST match that in
out   dx,al                   ; NIC_pstart

mov   dx,pspr                 ; end of receive buffer.
mov   al,eth_recv_buf_end     ; Value MUST match that in
out   dx,al                   ; NIC_pstop

mov   dx,NIC_cr               ; stop NIC activity
mov   al,eth_nic_stop
out   dx,al

mov   dx,NIC_dcr              ; local DMA transfers as
mov   al,eth_nic_DMA_config   ; 8 byte bursts
out   dx,al

mov   dx,NIC_rbcr0            ; remote DMA setup (remote
mov   al,eth_remote_DMA_lo    ; DMA not used, only local
out   dx,al                   ; used)

mov   dx,NIC_rbcr1            ; hi byte of # of bytes to
mov   al,eth_remote_DMA_hi    ; transfer during a remote
out   dx,al                   ; DMA operation

mov   dx,NIC_rcr              ; accept only good packets
mov   al,eth_packet_types     ;
out   dx,al

mov   dx,NIC_tcr              ; go into internal loopback
mov   al,eth_nic_mode         ; mode to finish programming
out   dx,al                   ; (see anomalies - p. 52)

mov   dx,NIC_bndy             ; overwrite protection rgtr.
mov   al,eth_bndy_start       ; (protects unread packets)
out   dx,al

mov   dx,NIC_pstart           ; start of receive queue
mov   al,eth_recv_buf_start
out   dx,al

mov   dx,NIC_pstop            ; end of receive queue
```

```
                mov   al,eth_recv_buf_end
                out   dx,al

                mov   dx,NIC_isr            ; clear interrupt status
                mov   al,eth_int_status
                out   dx,al

                mov   dx,NIC_imr            ; keep interrupts off
                mov   al,eth_ints_disabled  ;
                out   dx,al                 ;

                mov   dx,NIC_cr             ; access page 1 registers
                mov   al,eth_access_page_1
                out   dx,al

                mov   dx,phys_address_0      ; let NIC know its address
                mov   ax,cs
                mov   ds,ax
                mov   si,offset BOARD_ADDRESS ; from the prom
                cld
                mov   cx,6                   ; number of addresses to give
GIVE_ADDRESS:
                lodsb
                out   dx,al
                inc   dx
                loop  GIVE_ADDRESS          ; load all addresses

                mov   dx,NIC_curr           ; load current receive pointer
                mov   al,eth_recv_buf_start ; with pstart
                out   dx,al

                mov   dx,NIC_cr             ; access page 0 registers
                mov   al,eth_access_page_0
                out   dx,al

                mov   dx,NIC_cr             ; start NIC chip
                mov   al,eth_start_nic
                out   dx,al

                mov   dx,NIC_tcr            ; exit internal loopback mode
                mov   al,eth_exit_mode
                out   dx,al

                mov   ax,net_memory_seg     ; initialize LAN memory to
                mov   es,ax                 ; zeroes
                mov   cx,net_memory_size/2  ; in words
                xor   di,di                 ; start at begin of segment
                cld
                mov   ax,0000               ; initialization value
FILL:
                stosw
```

```
                loop  FILL

;
;  Now check our address against the known Ethernet addresses to determine
;  our processor ID
;
                mov  ax,cs
                mov  es,ax                 ; ds already = cs
                mov  bx,0                  ; init processor ID
                mov  di,offset NET_TABLE
                cld                        ; search direction = increment
Search:
                push di                    ; save start of current net addr
                mov  cx,3                  ; three words per address
                mov  si,offset BOARD_ADDRESS
                repe cmpsw
                pop  di                    ; restore current table index
                jz   Found
                add  di,8                  ; go to next index
                inc  bx                    ; count processor  id
                cmp  bx,NET_COUNT          ; see if all searched
                jnz  Search                ; loop back if more
;
;  If not found, it will return processor id = NET_COUNT
;
Found:
                mov  [PID],BX              ; record Processor ID
                ret                        ; done with Setup

                align 2
;
;  VALID PROCESSOR ID's Determined by Ethernet ADDRESS
;          0 - ALPHA
;          1 - BRAVO
;          2 - CHARLIE
;
PID             dw    ?                     ; Processor ID

BOARD_ADDRESS   db    6 dup (?)             ; holds board address

;
;   PROCESSOR STATION ADDRESS TABLE
;
NET_COUNT       equ     6                   ; number of processor on net

NET_TABLE       label   byte
       db      02H, 60H, 8CH, 47H, 63H, 55H,0,0  ; processor Bravo    1 VENUS
       db      02H, 60H, 8CH, 47H, 61H, 82H,0,0  ; processor Alpha    0 EARTH
       db      02H, 60H, 8CH, 48H, 51H, 60H,0,0  ; processor Charlie  2
       db      02H, 60H, 8CH, 58H, 35H, 68H,0,0  ; processor Delta    3
       db      02H, 60H, 8CH, 02H, 00H, 58H,0,0  ; processor Echo     4
```

```
        db      02H, 60H, 8CH, 44H, 52H, 09H,0,0  ; processor Foxtrot  5

cseg                    ends

                        END
```

```
        page    55,132
        TITLE   DA - Distributed Ada Module
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  FILE:  DA.ASM                                                     ;
;                                                                    ;
;  DA - Distributed Ada Module                                       ;
;                                                                    ;
;  Copyright(C) 1989, LabTek Corporation, Woodbridge, CT    USA      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                    ;
;  This code is code that would be part of the runtime system, but   ;
;  must be linked in to replace some part of the regular runtime     ;
;  routines.  It is Linked to the runtime via (hand) editing.        ;
;  Since the compiler does not supply information    ;
;  on the parameters in the code (it is implicitly maintained by     ;
;  the compiler amoung entry call/accept pairs), tables are placed
;  here to provide the information.
;                                                                    ;
;  Each packet header is statically formed and placed in this        ;
;  module to be reference by the TRANSMIT CONTROL PTR (TCP) used in  ;
;  the runtime call parameter list.  This reduces the overhead       ;
;  associated with packetizing the data.  These packet headers       ;
;  could be generated by the compiler/linker/distributor and         ;
;  optimally would be placed in the controller card memory at        ;
;  elaboration time so that loading of header data would be          ;
;  necessary.                                                        ;
;                                                                    ;
;                                                                    ;
;  Ver Date     Description                                          ;
;                                                                    ;
;  0.1 Nov-88 : Initial prototype                                    ;
;  0.2 Dec-89 : Enhanced to support error detection and dynamic      ;
;               configuration                                        ;
;                                                                    ;
;                                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        include DA_DEF.ASM

        .model  large

        public  Shut_down               ; prints out msg, and restarts
        public  COLD_START              ; NZ if this is cold start
        public  MODE_SELECT             ; Selected Operating Mode
        public  SYNC_PHASE              ; During startup to synchronize CPUs
        public  Print                   ; for sync printout
        public  MASTER                  ; for sync
        public  NUM_TARGETS             ; for sync Config set
        public  NUM_ROCKETS             ; for sync Config set

        extrn   Initialize:near         ; (rte)
```

```
        extrn   Ack_Check:near          ; (io)
        extrn   Sync:near               ; synchronize procedure
        extrn   VRTIF_APPLICATION:far   ; (vrtif)
        extrn   VRTIF_I8259:abs         ; (vrtif)
        extrn   DTCB_INIT:near          ; (dtcb)
        extrn   PID:word                ; processor id (Setup)

cseg    segment common
; BIOS Vectors
int10                           equ     40H
int16                           equ     58H
initial_imask                   equ     0FDH    ; mask off all but keyboard
da_base                         equ     3000H   ; segment for da runtime
upper_case                      equ     0DFH    ; mask for upper case characters
EGA_ROM_SEGMENT                 equ     0C000H
ROM_PRESENT                     equ     0AA55H

STACK_SIZE                      equ     200     ; bytes in local stack

MAX_ROCKETS                     equ     20      ; BDS maximum # rockets
MAX_TARGETS                     equ     50      ; BDS maximum # targets
MAX_MODE                        equ     4       ; BDS maximum mode value

ERROR_DELAY                     equ     70H     ; delay roughly 5 seconds
FLOPPY_STOP                     equ     0CH     ; Shuts off motors
FLOPPY_DIGITAL                  equ     3F2H    ; address of digital ctrl reg.


        assume  cs:cseg,ds:cseg,es:cseg

;
; The following jump table provides (static) control transfers from the
; Ada application code to the respective support code located here
;
        align   8               ; 00
        jmp     Restart         ; prior to elaboration
        align   8               ; 08
        jmp     Ack_Check       ; Check on Acknolwedgment of Messages
        align   8               ; 10
        jmp     Get_Master      ; Returns a boolean if this is the master
        align   8               ; 18
        jmp     Get_Rockets     ; Returns the number of Rockets Configured
        align   8               ; 20
        jmp     Get_Targets     ; returns the number of Targets Configured
        align   8               ; 28
        jmp     Get_Tasks       ; returns the number of Guide Tasks Configured
        align   8               ; 30
        test    word ptr cs:[EGA_PRESENT],0FFFFH
        jz      No_EGA
        jmp     dword ptr cs:[BIOS_VIDEO] ; vector to current EGA location
No_EGA: iret                    ; simply skip any EGA activity
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   Restart  to initialize the network hardware and configure the system
;
Restart:
        mov     cs:[AUTO],0             ; clear auto configure mode
Error_Restart:
        cli
        mov     dx,VRTIF_I8259+1        ; address of interrupt mask register
        mov     al,initial_imask        ; initial interrupt mask
        out     dx,al                   ; set mask
;
;  SETUP TEMPORARY STACK
;
        mov     ax,seg sseg
        mov     ss,ax
        mov     ax,STACK_SIZE
        mov     sp,ax
        call    clear                   ; @@ this is fix for compiler bug
;
;  SETUP DATA SEGMENT
;
        mov     ax,cs
        mov     ds,ax
;
;  CHECK COLD_START FLAG
;
        test    [COLD_START],0FFFH
        jz      Warm_start
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   COLD START ...  FIRST RELOCATE TO SEGMENT 3000
;
        mov     ax,da_base      ; first move stack segment
        mov     es,ax           ; save for later relocation of code/data
        mov     cx,cs
        sub     ax,cx           ; compute diff between load and base addr
        mov     cx,ss           ; now adjust stack segment
        add     ax,cx
        mov     ss,ax
        mov     cx,8000H
        mov     si,0FFFEH
        mov     di,0FFFEH
        std                     ; auto decrement
        rep     movsw
        cld
        mov     ax,da_base
        mov     ds,ax
        push    ax
        lea     ax,continue
        push    ax
```

```
        retf                    ; switch to 3000: segment
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Shut off floppy motor
;
continue:
        mov     al,FLOPPY_STOP
        mov     dx,FLOPPY_DIGITAL
        out     dx,al
;
;  Get BIOS Vectors
;
        mov     ax,0
        mov     es,ax           ; point to zero page
        mov     ax,es:[int10]
        mov     word ptr [BIOS_VIDEO],ax
        mov     ax,es:[int10+2]
        mov     word ptr [BIOS_VIDEO+2],ax
        mov     ax,es:[int16]
        mov     word ptr [BIOS_KB],ax
        mov     ax,es:[int16+2]
        mov     word ptr [BIOS_KB+2],ax
        mov     ax,EGA_ROM_SEGMENT
        mov     es,ax
        cmp     word ptr es:[0],ROM_PRESENT
        jnz     warm_start              ; if not present leave flag zero
        mov     [EGA_PRESENT],1         ; otherwise set flag

Warm_Start:
        call    Initialize              ; Initialize Ethernet Board
        mov     [SYNC_PHASE],1          ; set synchronization phase
        cmp     [PID],0                 ; see if we are the master
        jz      Master_CPU              ; go on if master
        jmp     Slave
;
;  if here, this is the master processor with a console
;
Master_CPU:
        mov     [MASTER],1              ; indicate this is the master
        test    [FAILURE],0FFFFH        ; see if display already setup
        jnz     skip_display
        call    Set_Display
skip_display:
        mov     [FAILURE],0     ; default is no failure (for next time)
        sti                     ; enable interrupts now for master mode
        test    [AUTO],0FFFFH     ; see if in auto reconfiguration mode
        jnz     Automatic
        call    Configure       ; perform Configuration
        jmp     TCB_setup
Automatic:
        call    Auto_Configure
```

```
        jmp     TCB_Setup       ; go on and setup tasks
;
;       if slave mode, do not attempt to configure, this is
;       done during the SYNC_PHASE sync procedure
;
Slave:
        mov     [MASTER],0      ; indicate THIS is not a master
        sti                     ; enable interrupts for slave mode
TCB_Setup:
;
;  Determine number of Guidance Tasks
;
        call    Sync            ; perform a synchronize
        mov     si,[MODE_SELECT] ; get selected mode
        dec     si              ; mode1 => offset 0
        add     si,si           ; double for word index
        mov     ax,GUIDE_TABLE[si]
        mov     [NUM_TASKS],ax  ; set number of tasks

        call    DTCB_Init       ; Initialize Task/Processor Directory
;
;  Go execute Application Code
;
        mov     [COLD_START],0  ; FINISHED WITH INITIALIZATION!
        jmp     VRTIF_APPLICATION

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   CONFIGURE - This routine sets the distributed system configuration
;
Configure:
        mov     si,offset ANNOUNCE
        call    Print
        call    Get_Char
        and     al,upper_case
        call    Put_Char                ; Echo Response
        cmp     al,'Y'
        jnz     Configure
rock00:
        mov     si,offset ROCKET_QUES
        call    Print
        call    Get_Num
        or      ax,ax
        jle     rock_error
        cmp     ax,MAX_ROCKETS
        jle     rock20
rock_error:
        mov     si,offset BAD_ROCKETS
        call    Print
        jmp     rock00
rock20:
```

```
        mov     [NUM_ROCKETS],ax

targ00:
        mov     si,offset TARGET_QUES
        call    Print
        call    Get_Num
        or      ax,ax
        jle     targ_error
        cmp     ax,MAX_TARGETS
        jle     targ20
targ_error:
        mov     si,offset BAD_TARGETS
        call    Print
        jmp     targ00
targ20:
        mov     [NUM_TARGETS],ax

mode00:
        mov     si,offset MODE_QUES
        call    Print
        call    Get_Num
        or      ax,ax
        jle     mode_error
        cmp     ax,MAX_MODE
        jle     mode20
mode_error:
        mov     si,offset BAD_MODE
        call    Print
        jmp     mode00
mode20:
        mov     [MODE_SELECT],ax ; establish mode

        mov     si,offset AUTO_QUES     ; see if auto reconfiguration desired
        call    Print
        call    Get_Char
        and     al,upper_case
        call    Put_Char
        cmp     al,'Y'
        jnz     auto_no
        mov     [AUTO],1
        mov     si,offset DELAY_QUES    ; if auto, check if delay desired
        call    Print
        call    Get_Char
        and     al,upper_case
        call    Put_Char
        cmp     al,'Y'
        jnz     delay_no
        mov     [DELAY],1
        jmp     config_done

auto_no:
```

```
        mov     [AUTO],0                ; shut off automatic mode
delay_no:
        mov     [DELAY],0

config_done:
        ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   AUTO_CONFIGURE - This routine sets the distributed system configuration
;        using an automatic allocation algorithm.
;
Auto_Configure:
        mov     [MODE_SELECT],1         ; for now, default to uniprocessor
        test    [DELAY],0FFFFH          ; see if we should delay
        jz      auto10                  ; if fast reconfigure requested

        lea     si,DELAY_MSG
        call    Print
        mov     ax,ERROR_DELAY
        call    DELAY_LOOP
auto10:
        ret

Delay_Loop:
        xor     cx,cx
delay_loop10:
        loop    delay_loop10
        dec     ax
        jnz     delay_loop10
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   SHUT_DOWN - Causes a message to be displayed indicating a network
;               error, and then jumps to Restart
;   This routine is entered only by the ACKCHECK service, therefore all
;   interrupts are currently disabled.
;
Shut_Down:
        cli
        mov     dx,VRTIF_I8259+1        ; address of interrupt mask register
        mov     al,initial_imask        ; initial interrupt mask
        out     dx,al                   ; set mask
        mov     ax,cs
        mov     ds,ax
        mov     [FAILURE],1             ; indicate we have a failure
        mov     dx,03CEH                ; @@@ straighten out display
        mov     al,5
        out     dx,al
        mov     dx,03CFH
```

```
        mov     al,0
        out     dx,al
        call    Set_Display             ; make sure display of all bits
        lea     si,NET_ERROR
        call    Print
        jmp     Error_Restart


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get_Master
Get_Master:
        push    ds
        mov     ax,cs
        mov     ds,ax
        mov     ax,[MASTER]
        pop     ds
        retf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get_Rockets
Get_Rockets:
        push    ds
        mov     ax,cs
        mov     ds,ax
        mov     ax,[NUM_ROCKETS]
        pop     ds
        retf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get_Targets
Get_Targets:
        push    ds
        mov     ax,cs
        mov     ds,ax
        mov     ax,[NUM_TARGETS]
        pop     ds
        retf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get_Tasks
Get_Tasks:
        push    ds
        mov     ax,cs
        mov     ds,ax
        mov     ax,[NUM_TASKS]
        pop     ds
        retf

;;;;;;;;;;;;;;;;;;;;;;;;
; PRINT - print string pointed to by SI until null
;
```

```
BIOS_WRITE      equ     09h             ; write color/attribute
PAGE_SELECT     equ     0
COLOR           equ     1fH             ; background blue, forground red
WINDOW_TOP      equ     0000H           ; row=0  col=0
WINDOW_BOTTOM   equ     184fH           ; row=24 col=79
SCROLL          equ     0601H           ; scroll up 1 row
CLEAR_DISPLAY   equ     0600H           ; scroll 0 = clear screen
GETCURSOR       equ     0300H
SETCURSOR       equ     0200H
cr              equ     000dH
lf              equ     000aH
bs              equ     0008H

index_reg       equ     03ceH           ; EGA index control register
display_select  equ        5
mode0           equ        0
mask_select     equ        8            ; select mask register
mask_bits       equ     0ffH            ; turn all bits on
sequence_reg    equ     03C4H
map_mask        equ     2


;
;  Set_Display - insures that the display bit mask has all bits turned on.
;                This is only necessary when switching from bit grahics
;                modes where typically only one bit is enabled.
;
Set_Display:
        test    word ptr [EGA_PRESENT],0FFFFH
        jnz     Set_Display10
        ret
Set_Display10:
        mov     ax,2
        pushf
        call    dword ptr [BIOS_VIDEO]

        mov     ax,CLEAR_DISPLAY
        mov     cx,WINDOW_TOP
        mov     dx,WINDOW_BOTTOM
        mov     bh,COLOR
        pushf                           ; push flags    (simulate INT 10H)
        call    dword ptr [BIOS_VIDEO]
        ret

;       mov     dx,index_reg
;       mov     al,display_select
;       out     dx,al
;       inc     dx                      ; point to data register
;       mov     al,mode0
;       out     dx,al
```

```
;       mov     dx,index_reg
;       mov     al,mask_select
;       out     dx,al
;       inc     dx                      ; point to data register
;       mov     al,mask_bits            ; set mask
;       out     dx,al
;
;       mov     dx,sequence_reg
;       mov     al,map_mask
;       out     dx,al
;       inc     dx
;       mov     al,mask_bits
;       out     dx,al
;
        ret
;
;  Print - write text pointed at by SI until null (0) is encountered
;
Print:
        cld
print10:
        lodsb
        or      al,al                   ; end of string?
        jz      print_end
        call    Put_Char                ; BIOS call preserves direction flag
        jmp     Print10
print_end:
        ret

;
;  Put_Char- writes character in AL on screen
;
Put_Char:
        test    word ptr [EGA_PRESENT],0FFFFH ; see if screen
        jnz     Put_char10
        ret

Put_Char10:
        push    ax
        cmp     al,cr                   ; Carriage return?
        jz      put_char_cr
        cmp     al,lf                   ; line feed
        jz      put_char_lf
        cmp     al,bs                   ; back space
        jz      put_char_bs

        mov     ah,BIOS_WRITE           ;
        mov     bh,PAGE_SELECT          ; select page
        mov     bl,COLOR                ; set color
        mov     cx,1                    ; 1 character
        pushf                           ; push flags    (simulate INT 10H)
```

```
        call    dword ptr [BIOS_VIDEO]
        mov     ax,GETCURSOR
        mov     bh,PAGE_SELECT
        pushf
        call    dword ptr [BIOS_VIDEO]

        mov     ax,SETCURSOR
        mov     bh,PAGE_SELECT
        inc     dl                      ; move cursor over
        pushf
        call    dword ptr [BIOS_VIDEO]
        jmp     Put_Char_end

put_char_cr:
        mov     ax,GETCURSOR
        mov     bh,PAGE_SELECT
        pushf
        call    dword ptr [BIOS_VIDEO]

        mov     ax,SETCURSOR
        mov     bh,PAGE_SELECT
        mov     dl,0                    ; reset column
        pushf
        call    dword ptr [BIOS_VIDEO]
        jmp     Put_char_end

put_char_lf:
        mov     ax,GETCURSOR            ; see if at bottom of screen
        mov     bh,PAGE_SELECT
        pushf
        call    dword ptr [BIOS_VIDEO]

        cmp     dh,24                   ; at bottom?
        jz      put_char_lf10
        inc     dh                      ; if not at bottom of screen, just go
        mov     ax,SETCURSOR            ; down 1 more line
        mov     bh,PAGE_SELECT
        pushf
        call    dword ptr [BIOS_VIDEO]
        jmp     Put_char_end

put_char_lf10:
        mov     ax,SCROLL               ; if at bottom, then scroll
        mov     cx,WINDOW_TOP
        mov     dx,WINDOW_BOTTOM
        mov     bh,COLOR
        pushf                           ; push flags    (simulate INT 10H)
        call    dword ptr [BIOS_VIDEO]
        jmp     Put_char_end

put_char_bs:
```

```
        mov     ax,GETCURSOR
        mov     bh,PAGE_SELECT
        pushf
        call    dword ptr [BIOS_VIDEO]

        mov     ax,SETCURSOR
        mov     bh,PAGE_SELECT
        or      dl,dl                   ; see if already at left margin
        jz      put_char_bs2
        dec     dl                      ; adjust column
        pushf
        call    dword ptr [BIOS_VIDEO]

put_char_bs2:
        j.p     Put_char_end

put_char_end:
        pop     ax
        ret

GET_KB          equ     0               ; read character (synchronous)

Get_Char:
        mov     ax,GET_KB
        pushf
        call    dword ptr [BIOS_KB]
        ret
;
;  Accepts a number from console
;       Returns with AX having value (0 if blank line entered)
;
Get_Num:
        call    Get_Line
        mov     si,offset LINE_BUFF
        mov     ax,0            ; init value
        mov     bx,10           ; decimal numbers
        mov     ch,0            ; high byte
Get_num10:
        mov     cl,[si]
        inc     si
        cmp     cl,cr           ; see if end of line
        jz      Get_num20
        cmp     cl,' '          ; also terminate on space
        jz      Get_num20
        cmp     cl,'0'
        jl      get_num_error
        cmp     cl,'9'
        jg      get_num_error
        mul     bx
        and     cl,0fH
        add     ax,cx
```

```
        jmp     Get_num10

Get_num_error:
        mov     si,offset Input_Error
        call    Print
        jmp     Get_Num

Get_num20:
        ret
;
;   Get_Line - fetches line from keyboard until <CR> is entered
;       returns with line in LINE_BUFF terminated by <CR>
;
Get_Line:
        mov     si,offset LINE_BUFF
get_line10:
        cmp     si,offset END_OF_LINE
        jz      get_line_cr             ; force a <CR>
        call    Get_Char
        cmp     al,bs                   ; backspace?
        jz      get_line_bs
        cmp     al,cr
        jz      get_line_cr
        mov     [si],al
        inc     si
        call    Put_Char
        jmp     get_line10

get_line_bs:
        cmp     si,offset LINE_BUFF
        jz      get_line10              ; do nothing if at begin of line
        call    Put_Char
        mov     al,' '
        call    Put_Char
        mov     al,bs
        call    Put_Char
        dec     si                      ; back up buffer pointer
        jmp     get_line10

get_line_cr:
        mov     al,cr
        mov     [si],al
        call    Put_Char
        mov     al,lf
        call    Put_Char
        ret

;
; CLEAR - routine to zero some of memory to compensate for code
;         generator error.
Clear:
```

```
        mov     ax,951bH
        mov     es,ax
        xor     ax,ax
        mov     cx,ax
        mov     di,ax
        cld
        rep     stosw
        ret

        align   4
;
;  BIOS Actual Routine Addresses
;
BIOS_VIDEO      dd      ?
BIOS_KB         dd      ?
EGA_PRESENT     dw      0               ; default is not present

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   CONFIGURATION INFORMATION SUPPLIED TO THE APPLICATION           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MASTER          dw      ?               ; NZ if this is the master CPU
NUM_ROCKETS     dw      ?               ; max number of rockets to launch
NUM_TARGETS     dw      ?               ; max number of targets to generate
NUM_TASKS       dw      ?               ; number of GUIDANCE tasks
;
;   Indexed by mode
;
GUIDE_TABLE     dw      1               ; mode 1 = 1 guide task
                dw      1               ; mode 2 = 1 guiqe task
                dw      1               ; mode 3 = 1 guide task
                dw      2               ; mode 4 = 2 guide tasks

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   OPERATION CONTROL VARIABLES                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

COLD_START      dw      1               ; cold start=1 if first time through
SYNC_PHASE      dw      ?               ; initial synchronization phase
MODE_SELECT     dw      ?               ; selected operating mode
AUTO            dw      0               ; default is not automatic mode
DELAY           dw      0               ; if a delay before restart is desired
FAILURE         dw      0               ; if a network failure occured

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   TEXT INPUT/OUTPUT DEFINITIONS                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LINE_BUFF       db      256 dup (?)
END_OF_LINE     equ     $-1

ANNOUNCE        db      cr,lf,lf
```

```
                db      '                  THE    BORDER    DEFENSE'
                db      cr,lf,lf
                db      '  D I S T R I B U T E D    A d a    '
                db      'C O N F I G U R A T I O N    M E N U',cr,lf,lf,lf
                db      'Type ''Y'' to continue: ',0
ROCKET_QUES     db      cr,lf,lf,'Enter Number Of Rockets=>  ',0
TARGET_QUES     db      cr,lf,lf,'Enter Number of Targets=>  ',0
MODE_QUES       db      cr,lf,lf,'SELECT MODE:',cr,lf
                db      '             1 = Single    Processor',cr,lf
                db      '             2 = Dual (AB) Processor',cr,lf
                db      '             3 = Dual (AC) Processor',cr,lf
                db      '             4 = Triple    Processor',cr,lf
                db      'MODE => ',0
AUTO_QUES       db      cr,lf,lf,'Automatic Reconfiguration? (Y)    : ',0
DELAY_QUES      db      cr,lf,lf,'Delay before Reconfiguration? (Y) : ',0
BAD_ROCKETS     db      cr,lf,'Out Of Range!, Rockets must be between 1 and 20'
                db      cr,lf,lf,'Reenter: ',0
BAD_TARGETS     db      cr,lf,'Out Of Range!, Targets must be between 1 and 50'
                db      cr,lf,lf,'Reenter: ',0
BAD_MODE        db      cr,lf,'Out Of Range!, Mode must be between 1 and 4'
                db      cr,lf,lf,'Reenter: ',0
INPUT_ERROR     db      cr,lf,'Invalid Number'
                db      cr,lf,'Reenter: ',0

NET_ERROR       db      lf,lf,' N E T W O R K   T R A N S M I S S I O N   '
                db      'E R R O R   D E T E C T E D !',cr,lf,lf,0

DELAY_MSG       db      'SYSTEM WILL RESTART IN FIVE SECONDS...',cr,lf,lf,0

cseg      ends


sseg      segment STACK
          db      STACK_SIZE dup (0)
sseg      ends

          end
```

Distributed Issues Final Report